Introduction to Distributed Systems

527950-1
Fall 2019
9/19/2019
Kyoung Shin Park
Applied Computer Engineering
Dankook University

Chapter 1. Characterization of Distributed Systems

From Coulouris, Dollimore, Kindberg and Blair
Distributed Systems: Concepts and Design
Edition 5, © Addison-Wesley 2012

Overview

- Definition of Distributed Systems
- Characteristics of Distributed Systems
- Motivation of Distributed Systems
- Examples of Distributed Systems
- Trends of Distributed Systems
- Challenges (Issues)

Definition of Distributed Systems

- **"a collection of independent computers that appears to its users as a single coherent system"**(Tanenbaum)
- "a collection of autonomous computers linked by a computer network with distributed system software"(CDK)
- "a collection of processors interconnected by a communication network in which each processor has its own local memory and other peripherals and the communication between any two processors of the system takes place by message passing over the communication network"(Sinha)
- **"one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages" (CDK)**

Definition of Distributed Systems

- A collection of independent computers that appears to users as a single system - a virtual uniprocessor
  - Users do not know *(or care)* where *(on what machine)* files are located and where a job is executed
- A distributed system is made of several computers which
  - Have no shared memory
  - Have no shared clock
  - Communicate with each other via messages
  - Each computer has its own operating system

Definition of Distributed Systems

- Computer networks vs. Distributed systems
  - Computer network - the autonomous computers are explicitly visible
  - Distributed system - existence of multiple autonomous computers is transparent
  - However, many problems in common

Characteristics of Distributed Systems

- **Concurrency** of components
  - Concurrent program execution
  - Coordination of concurrently executing programs that share resources
- **Independent failures** of components
  - Each component of the system can **fail independently**, leaving the **others still running**
  - Faults in the network result in the isolation of the computers that are connected to it but that doesn't mean that they stop running
  - Failure of a computer, or the unexpected termination of a program (a crash), is **not immediately made known to the other components** with which it communicates

Characteristics of Distributed Systems

- Lack of a **global clock**
  - Close coordination often depends on a shared idea of the time at which the programs' actions occur
  - There are **limits to the accuracy** with which the computers in a network can synchronize their clocks
  - There is **no single global notion of the correct time**

Motivation of Distributed Systems

- **Resource sharing**
  - Hardware components such as disks and printers
  - Software-defined entities such as files, DBs and data objects of all kinds. It includes the stream of video frames and the audio connection.
- **Functional distribution** - computers have different functional capabilities
  - Client/server
  - Data gathering/data processing
  - Sharing of resources with specific functionalities
- **Load distribution/balancing**
  - Assign tasks to processors such that the overall system performance is optimized

Motivation of Distributed Systems

- **Replication of processing power**
  - Distributed systems consisting of collections of microcomputers may have processing powers that no supercomputer will ever achieve => Shorter response time; higher throughput
- **Economics**
  - Collections of microprocessors offer a better price/performance than large mainframes
- Improved reliability and availability
  - If one component goes down, the system does not
- Modular expandability
- Inherently distributed applications
  - Airline reservations; Bank ATMs
- Better flexibility

Examples of Distributed Systems

- Distributed systems encompass many of the *most significant technological developments* of recent years
- An initial insight into the wide range of applications in use today
  - from relatively localized systems (as found, for example, in a car or aircraft) to global scale systems involving millions of nodes,
  - from data-centric services to processor intensive tasks,
  - from systems built from very small and relatively primitive sensors to those incorporating powerful computational elements,
  - from embedded systems to ones that support a sophisticated interactive user experience, and so on.

Examples of Distributed Systems

- Selected range of key commercial or social application sectors highlighting some of the associated established or emerging uses of distributed systems technology

Finance and commerce	eCommerce (e.g. Amazon and eBay, PayPal, Online banking and trading)
The information society	Web information and Search engines (ebooks, Wikipedia); Social networking (Facebook and MySpace).
Creative industries and entertainment	Online gaming, Music and film in the home, User-generated content (e.g. YouTube, Flickr)
Healthcare	Health informatics (online patient records, monitoring patients)
Education	e-learning, Virtual learning environments; Distance learning
Transport and logistics (location-aware service)	GPS in route finding systems, map services (Google Maps, Google Earth)
Science	The Grid as an enabling technology for collaboration between scientists
Environmental management	Sensor technology (to monitor earthquakes, floods or tsunamis)

Examples of Distributed Systems

- **Web search**
  - Analyze the entire web content and then carry out sophisticated processing on this enormous database
    - Represents a major challenge for distributed systems design
  - **Google** has put significant effort into the design of a sophisticated distributed system infrastructure
    - This represents one of the largest and most complex distributed systems installations in the history of computing
    - Highlights of this infrastructure include:
      1. an underlying physical infrastructure consisting of very large numbers of networked computers located at **data centers** all around the world;
      2. a distributed file system designed to support very large files
      3. an associated structured distributed storage system that **offers fast access to very large datasets**;
      4. a lock service that offers distributed system functions such as **distributed locking and agreement**;
      5. a programming model that supports the management of **very large parallel and distributed computations** across the underlying physical infrastructure.

Examples of Distributed Systems

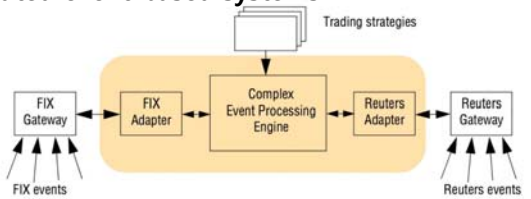
- **Massively Multiplayer Online Games (MMOGs)**
  - Very large numbers of users interact through the Internet with a persistent virtual world
    - Over 50,000 simultaneous online players (and the total number of players perhaps ten times this figure)
    - The need for fast response times to preserve the user experience of the game
    - The real-time propagation of events to the many players and maintaining a consistent view of the shared world.

Examples of Distributed Systems

- **Massively Multiplayer Online Games (MMOGs)**
  - A number of solutions
    - A client-server architecture where a single copy of the state of the world is maintained on a **centralized server** and accessed by client programs
    - More distributed architectures where the **universe is partitioned** across a (potentially very large) number of servers that may also be geographically distributed
    - Looking at more radical architectures that are not based on client-server principles but rather adopt **completely decentralized approaches** based on peer-to-peer technology where every participant contributes resources (storage and processing) to accommodate the game

Examples of Distributed Systems

- **Financial trading**
  - Real-time access to a wide range of information sources
    - For example, current share prices and trends, economic and political developments
  - Emphasis is on
    - **The communication and processing of items of interest**, known as **events**,
    - With the need also to deliver events reliably and in a timely manner
    - To potentially very large numbers of clients who have a stated interest in such information items.
  - **Distributed event-based systems**

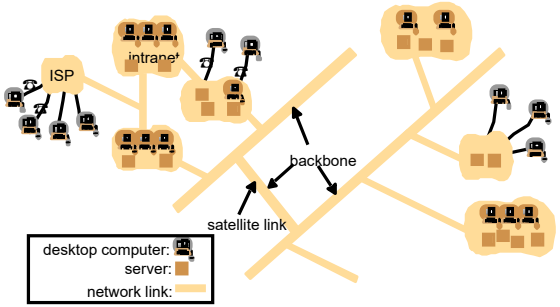


Trends in Distributed Systems

- The emergence of **pervasive networking technology**
  - Internet is a vast interconnected collection of computer networks of many different types, with the range of types increasing all the time
  - A wide range of wireless communication technologies such as WiFi, Bluetooth and mobile phone networks
  - A **pervasive resource and devices** can be connected at any time and in any place

Trends in Distributed Systems

- **The Internet** is also a very large distributed system
  - Make use of services such as the WWW, email and file transfer
  - Intranets – **subnetworks** operated by companies and other organizations and typically protected by firewalls
  - Internet Service Providers (ISPs) are companies that provide broadband links and other types of connection to individual users and small organizations
  - The intranets are linked together by backbones



Trends in Distributed Systems

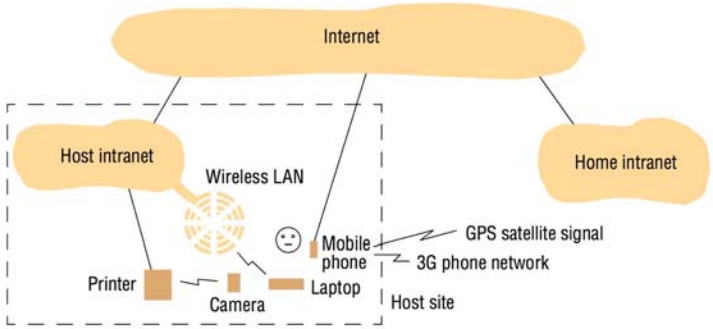
- The emergence of **ubiquitous computing** coupled with the desire to support **user mobility** in distributed systems
  - Device miniaturization and **wireless networking**
    - Laptop computers
    - Handheld devices
    - Wearable devices
    - Devices embedded in appliances
  - **Mobile computing**
    - Computing tasks while the user is on the move
    - **Location-aware** or **context-aware computing**

Trends in Distributed Systems

- **Ubiquitous computing**
  - Harnessing of many small, cheap computational devices that are present in users' physical environments, including the home, office and even natural settings.
  - Devices will become so **pervasive** in everyday objects that are scarcely noticed
- Ubiquitous and mobile computing overlap, but they are **distinct**
  - Ubiquitous but not mobile - Ubiquitous computing could benefit users while they remain in a single environment such as the home or a hospital
  - Mobile but not ubiquitous - Mobile computing has advantages even if it involves only conventional devices

Trends in Distributed Systems

- **Portable and handheld devices** in a distributed system



- Spontaneous interoperation
  - Associations between devices are routinely created and destroyed
- Service discovery
  - Associating the device with **suitable local services**

Trends in Distributed Systems

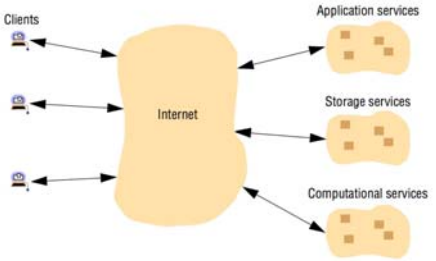
- The increasing demand for **multimedia services**
  - Crucial characteristic of continuous media types
    - **Real-time relationships** between elements of a media type
  - A wide range of new (multimedia) services and applications
  - Multimedia applications such as webcasting place considerable demands on the underlying distributed infrastructure in terms of
    - Providing support for a range of encoding and encryption formats
    - Providing a range of mechanisms to ensure that the desired quality of service(QoS) can be met
    - Providing associated resource management strategies
    - Providing adaptation strategies to deal with the inevitable situation

Trends in Distributed Systems

- The view of distributed systems as a commodity or **utility**
  - Resources are provided by appropriate service suppliers and effectively rented rather than owned by the end user
  - Applies to both physical resources and more logical services
  - Physical resources such as **storage** and **processing**
    - From remote storage facility to data centers
    - OS virtualization is a key enabling technology for this approach
  - Software services
    - Email and distributed calendars

Trends in Distributed Systems

- **Cloud computing**
  - Defined as a set of Internet-based application, storage and computing services
  - A view of everything as a service
  - Generally implemented on **cluster computers** to provide the necessary scale and performance required by such services
  - A **cluster computer** is a set of interconnected computers that cooperate closely to provide a single, integrated high performance computing capability



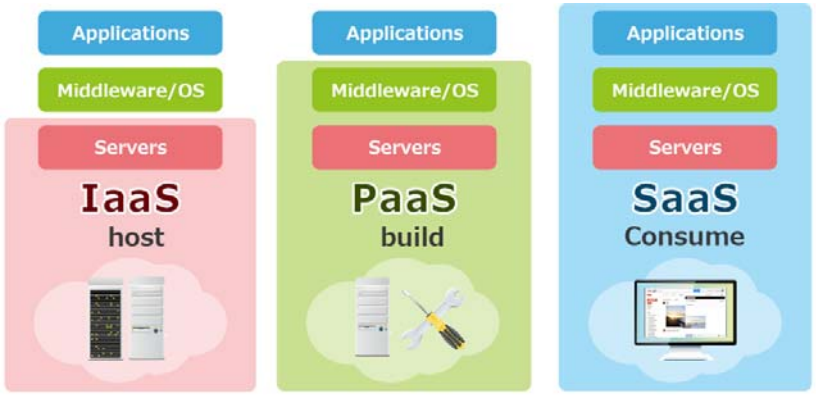
Cloud Computing

- Information technology (IT) paradigm
  - enables ubiquitous access to shared pools of configurable system resources and higher-level services
  - can be rapidly provisioned with minimal management effort, often over the Internet
- **Infrastructure** as a service (**IaaS**)
- **Platform** as a service(**PaaS**)
- **Software** as a service (**SaaS**)

Cloud Computing

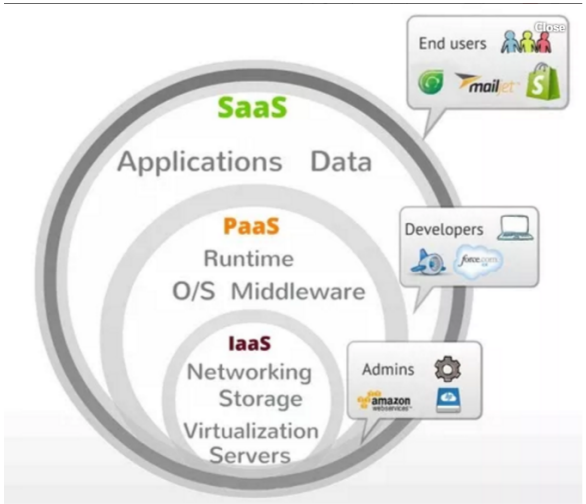
- **"Infrastructure as a service" (IaaS)**
  - refers to online services that provide high-level APIs used to dereference various low-level details of underlying network infrastructure like physical computing resources, location, data partitioning, scaling, security, backup etc.
- **Platform as a Service (PaaS)**
  - category of cloud computing services that provides a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app
- **Software as a service (SaaS)**
  - access to application software
  - use the provider's applications running on a cloud infrastructure
  - The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface.

Cloud Computing



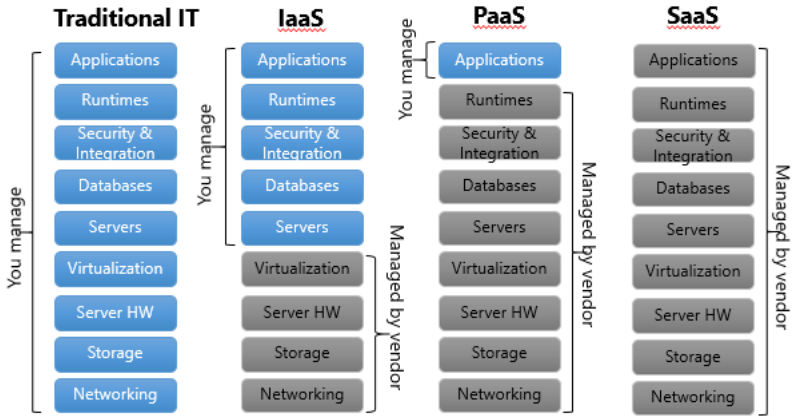
https://medium.com/@Albihany/true-cloud-story-about-iaas-paas-saas-47cfea883271

Cloud Computing



http://blog.webspecia.com/cloud/iaas-paas-saas-explained-examples-comparison

Cloud Computing



http://www.mazikglobal.com/blog/wp-content/uploads/2014/06/Cloud-Service-Models.png

Problems with Distributed Systems

- All communication is done by **message passing** - i.e.
  - All coordination is decentralized
  - There is a lack of global information
  - There may be replication of data
- How can failures be detected and recoveries made?

Goals of a Distributed System

- Goals of a distributed system (Tanenbaum & van Steen)
  - Connecting users and resources
  - Transparency
  - Openness
  - Scalability

Goals of a Distributed System

- Characteristics of a distributed system (Galli)
  - Shared resources
  - Openness
  - Concurrency
  - Scalability
  - Fault tolerance
  - Transparency

Goals of a Distributed System

- Challenges of a distributed system (Coulouris, Dollimore and Kindberg)
  - Heterogeneity
  - Openness
  - Security
  - Scalability
  - Failure handling
  - Concurrency
  - Transparency
  - Quality of Service

Heterogeneity

- A system is **heterogeneous** if it is composed of dissimilar hardware and software.
  1. Networks
     - Their differences are masked by the fact that all of the computers attached to them use the Internet protocols to communicate with one another – **IP-based networks**
     - The Internet protocols are implemented over a variety of different networks
  2. Computer hardware
     - CPUs - byte ordering of integers(big-endian vs. little-endian)
  3. Operating systems
     - Do not necessarily all provide the same API to Internet protocols
  4. Programming languages
     - Use different representations for characters and data structures
  5. Implementations by different developers
     - **Use common standards**, for example, for network communication and the representation of primitive data items and data structures in messages

Heterogeneity

- Heterogeneity can be contrasted with **portability**:
  - A program that works in a heterogeneous environment must deal with various hardware and software components at the same time,
  - Whereas a portable program must run on different systems at different times.
- A related notion is **interoperability**. It denotes the ability of different components, possibly from different vendors, to interact.

Heterogeneity

- The goal of heterogeneity is to have
  - different operating systems
  - different computer hardware
  - different networks
  - different programming languages
- All working together to form a single distributed system
- **Communication protocols** can be used to mask the network differences
- **Middleware**, "an additional layer of software between the applications and the network OS" (Tanvan02) can be used to handle other differences

Heterogeneity

- **Middleware**
  - A software layer that provides a **programming abstraction** as well as **masking the heterogeneity** of the underlying networks, hardware, operating systems and programming languages.
  - Provides a **uniform computational model** for use by the programmers of servers and distributed applications - remote object invocation, remote event notification, remote SQL access and distributed transaction processing
  - Common Object Request Broker (CORBA)
  - Java Remote Method Invocation (RMI)
- **Mobile code**
  - Program code that can be transferred from one computer to another and run at the destination
  - Java applet, JavaScript
  - Virtual machine approach provides a way of making code executable on a variety of host computers

Openness

- A characteristic that enables systems to be **extended to meet new** application requirements and user needs
  - i.e., characteristic that determines whether the system can be extended and re-implemented in various ways
- Determined primarily by
  - The degree to which new resource-sharing services can be added and
  - Be made available for use by a variety of client programs
- Advantage
  - Extensible at the hardware level by the addition of computers to the network and at the software level by the introduction of new services and the reimplementation of old ones, enabling application programs to share resources
  - Independent from individual vendors

Openness

- Characteristics
  - Achieved by **specifying and documenting the key software interfaces** of a system and making them available to software developers; i.e. **the interfaces should be publicly available** to ease inclusion of new components
  - Based on the provision of a uniform communication mechanism and published interfaces for access to shared resources.
  - Can be constructed from heterogeneous hardware and software, possibly from different vendors.

Security

- Confidentiality
  - Protection against disclosure to unauthorized individuals
  - Make sure the identity of the user
- Integrity
  - Protection against alteration or corruption
  - Send sensitive information in a message over a network in a secure manner – encryption technique
- Availability
  - Protection against interference with the means to access the resources
  - Denial of service(DoS) attacks - wish to disrupt a service for some reason

Security

- Firewall
  - A barrier around an intranet, restricting the traffic that can enter and leave
  - Not deal with ensuring the appropriate use of resources by users within an intranet, or with the appropriate use of resources in the Internet
- Security of mobile code
  - Executable program, such as an electronic mail attachment

Scalability

- Scalability refers to the ability of a distributed system to grow without users or applications knowing or being affected
- How can this be done
  - Without a global clock or memory?
  - Without a global state?
  - By avoiding any centralized components in the distributed system
    - Software
    - Hardware
    - Algorithms
  - By basing decisions
    - Solely on locally-known information
  - By recognizing that
    - Any component could go down at any time
    - And being able to continue anyway

Scalability

- **Challenges** for the design of **scalable distributed system**
  - Controlling the cost of physical resources
    - As the demand for a resource grows, it should be possible to extend the system, at reasonable cost, to meet it
  - Controlling the performance loss
    - Algorithms that use hierarchic structures scale better than those that use linear structures
  - Preventing software resources running out
    - IPv4(32 bit Internet address) vs IPv6(128 bit Internet address)
    - It is difficult to predict the demand that will be put on a system years ahead.
    - Overcompensating for future growth may be worse than adapting to a change when we are forced to
  - Avoiding performance bottlenecks – load balancing
    - Algorithms should be decentralized to avoid having performance bottlenecks
    - Caching and replication may be used to improve the performance of resources that are very heavily used

Failure Handling

- Need **fault tolerance** if it is able to continue processing when one or more components of the system fail
- For distributed system to be fault tolerant, it must be able
  - To detect errors, faults, threats, or other failures
  - To tolerate the failures (i.e., not stumble or crash)
  - To mask the failures (i.e., hide them from the user)
  - To recover from the failures
- Both redundancy and decentralization support fault tolerance
- When faults occur
  - May produce incorrect results or
  - May stop before they have completed the intended computation
- Failures in a distributed system are partial
  - some components fail while others continue to function

Failure Handling

- Techniques for dealing with failures
  - Detecting failures
    - Checksums to detect corrupted data; difficult to detect some other failures, such as a remote crashed server in the Internet
  - Masking failures - hidden or made less severe
    - Messages can be retransmitted
    - File data can be written to a pair of disks
  - Recovery from failures
    - The state of permanent data can be recovered or '**rolled back**' after a server has crashed
    - The computations will be incomplete when a fault occurs, and the permanent data that they update may not be in a consistent state
  - Redundancy
    - Tolerate failures by the use of redundant components
    - Multiple routes in network – multiple disjoint paths
    - A database may be replicated in several servers
    - The design of effective techniques for keeping replicas of rapidly changing data up-to-date without excessive loss of performance is a challenge

Concurrency

- Clearly one of the goals of a distributed system is to **share resources** whether data, files, equipment, or machine cycles.
  - This is similar to what is expected of a time-sharing system.
  - However, in a distributed system, there are many processors.
- This means that any user can be active at any time and on any processor. This also means that any resource can have more than one **concurrent** request simultaneously.
  - Several clients will attempt to access a shared resource at the same time, producing inconsistent results
  - Race condition
- All **mutual exclusion** and **deadlock problems** are more to be considered, but with the caveat that more than one processor can be requesting any given shared resource.
  - For an object to be safe in a concurrent environment, its operations must be synchronized in such a way that its data remains consistent – achieved by techniques such as semaphores

Transparency

- Network, Access, Location Transparency
- Name Transparency
- Concurrency, Parallelism Transparency
- Replication Transparency
- Migration (or Relocation), Persistence Transparency
- Failure Transparency
- Performance, Scaling Transparency
- Revision, Size Transparency

Network, Access, Location Transparency

- Network transparency = access + location transparency
- Access transparency
  - Enables local and remote information objects to be accessed using identical operations, e.g. Distributed File Systems
  - This means that whether some processors in the distributed system are Windows machines, Unix machines, or Macs, whether they are Big or Little Endian machines
  - The user is able to **access objects** located on them **in exactly the same manner**
- Location transparency
  - Enables information objects to be accessed without knowledge of their physical or network location, e.g. which building or IP address
  - URLs are location-transparent, but not mobility-transparent

Name Transparency

- Name transparency
  - The distributed system incorporates a **global naming scheme**
  - Objects (files, resources) are not tied to given nodes or sites by name
  - Name transparency assists migration, access, and location transparencies

Concurrency, Parallelism Transparency

- Concurrency transparency
  - Enables several processes to operate concurrently using shared information objects without interference between them
  - Permits efficient use of shared resources
  - Allows no interference between processes sharing resources
- Parallelism transparency
  - Permits parallel activities without users knowing how, where, and when these activities are carried out in the system

Replication Transparency

- Replication transparency
  - Enables multiple instances of information objects to be used to increase reliability and performance without knowledge of the replicas by users or application programmers
  - This means that there may be multiple copies of files scattered over the entire distributed system

Migration, Persistence Transparency

- Migration (or Relocation) transparency
  - Allows the movement of information objects within a system without affecting the operation of users or application programs
  - This means that both resources and processes can migrate without users knowing and be accessed while being relocated
- Persistence transparency
  - Refers to the type of memory where files are located
  - Specifically, whether or not that memory is stable or volatile

Failure Transparency

- Failure transparency
  - Enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components
  - This means if a site goes down, it should be unapparent to other sites or users and work continues

Performance, Scaling Transparency

- Performance transparency
  - Allows the system to be reconfigured to improve performance as loads vary
- Scaling transparency
  - Allows the system and applications to **expand** in scale without change to the system structure or the application algorithms

Size, Revision Transparency

- Size transparency
  - Allows incremental growth of a system without the user's awareness
  - Clearly, this is a form of scaling transparency
- Revision transparency
  - Software revisions of the system are not visible to users
  - This is also a form of scaling transparency

Quality of Service (QoS)

- Ability to meet service requirements for applications
- Its achievement depends upon the availability of the necessary computing and network resources at the appropriate times.
  - A requirement for the system to provide guaranteed computing and communication resources that are sufficient to enable applications to complete each task on time
  - Each critical resource must be reserved by the applications that require QoS
- **Guaranteed service vs. best-effort service**
- The main nonfunctional properties of systems that affect the QoS experienced by clients and users
  - Reliability, security and performance
  - Adaptability to meet changing system configurations and resource availability