Introduction to Distributed Systems

PhD Jorge E. Villaverde – Desarrollo de Aplicaciones Cliente-Servidor

What is a distributed system?

- A distributed system is a collection of independent computers that appear to the users of the system as a single system.
- Examples:
  - Network of workstations
  - Distributed manufacturing system (e.g., automated assembly line)
  - Network of branch office computers

Concepts

- **A program**: is the code you write.
- **A process**: is what you get when you run it.
- **A message**: is used to communicate between processes.
- **A packet**: is a fragment of a message that might travel on a wire.
- **A protocol**: is a formal description of message formats and the rules that two processes must follow in order to exchange those messages.
- **A network**: is the infrastructure that links computers, workstations, terminals, servers, etc. It consists of routers which are connected by communication links.
- **A component**: can be a process or any piece of hardware required to run a process, support communications between processes, store data, etc.
- **A distributed system**: is an application that executes a collection of protocols to coordinate the actions of multiple processes on a network, such that all components cooperate together to perform a single or small set of related tasks.

Why build a distributed system?

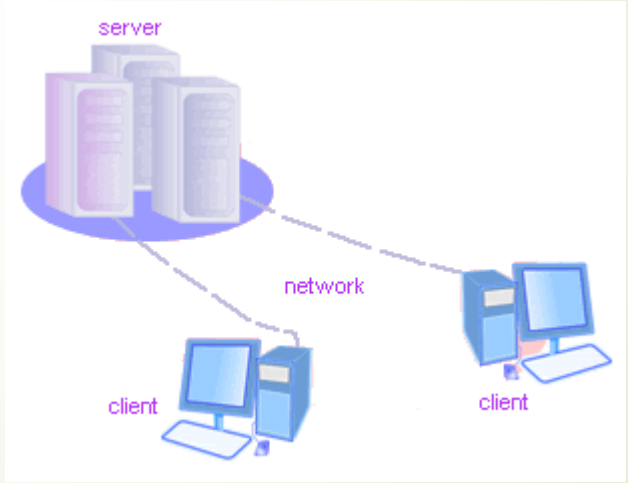
- **Fault-Tolerant**: It can recover from component failures without performing incorrect actions.
- **Highly Available**: It can restore operations, permitting it to resume providing services even when some components have failed.
- **Recoverabl**e: Failed components can restart themselves and rejoin the system, after the cause of failure has been repaired.
- **Consistent**: The system can coordinate actions by multiple components often in the presence of concurrency and failure. This underlies the ability of a distributed system to act like a non-distributed system.
- **Scalable**: It can operate correctly even as some aspect of the system is scaled to a larger size. For example, we might increase the size of the network on which the system is running. This increases the frequency of network outages and could degrade a "non-scalable" system. Similarly, we might increase the number of users or servers, or overall load on the system. In a scalable system, this should not have a significant effect.
- **Predictable Performance**: The ability to provide desired responsiveness in a timely manner.
- **Secure**: The system authenticates access to data and services.

8 Fallacies of Distributed Systems

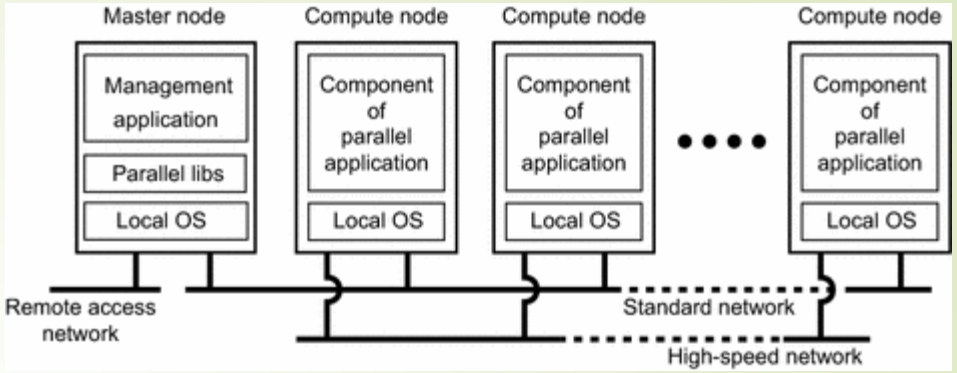
- **The network is reliable.**
- **Latency is zero.**
- **Bandwidth is infinite.**
- **The network is secure.**
- **Topology doesn't change.**
- **There is one administrator.**
- **Transport cost is zero.**
- **The network is homogeneous.**

Client-Server applications

- the *server* provides some service, such as processing database queries or sending out current stock prices.
- The *client* uses the service provided by the server, either displaying database query results to the user or making stock purchase recommendations to an investor.



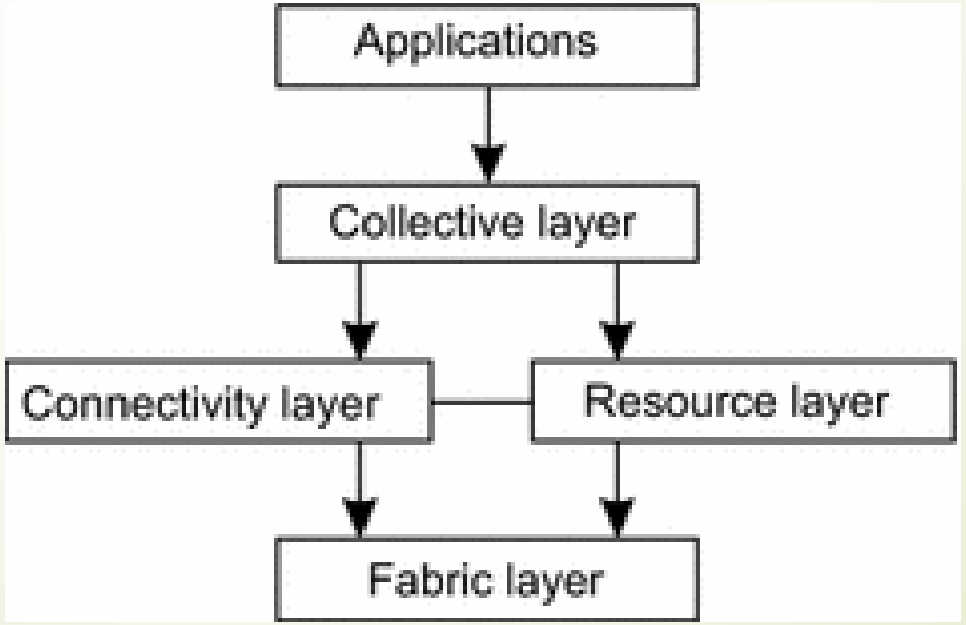
Cluster computing



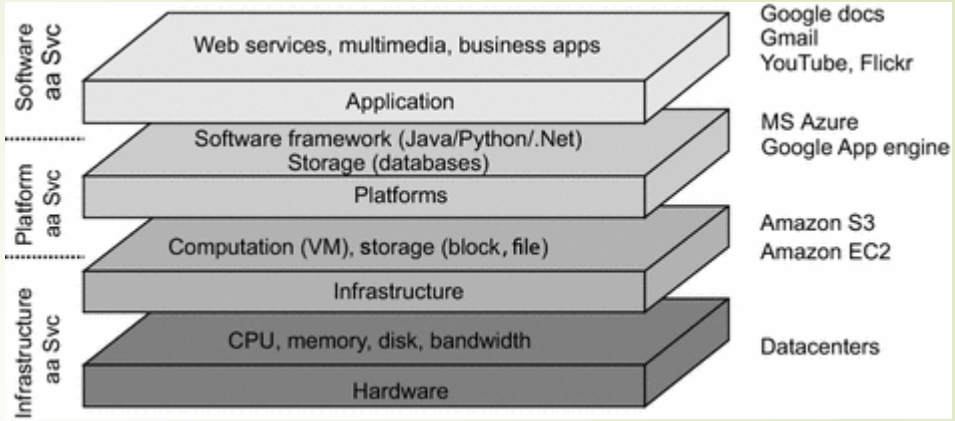
- cluster computing is used for parallel programming in which a single (compute intensive) program is run in parallel on multiple machines.
- Each cluster consists of a collection of compute nodes that are controlled and accessed by means of a single master node.
- The master typically handles the allocation of nodes to a particular parallel program, maintains a batch queue of submitted jobs, and provides an interface for the users of the system.

Grid computing

- A characteristic feature of traditional cluster computing is its homogeneity.
- In Grid computing no assumptions are made concerning similarity of hardware, operating systems, networks, administrative domains, security policies, etc.

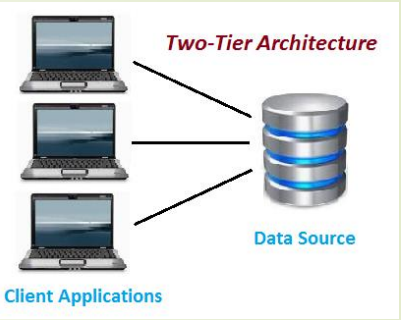


Cloud computing



- **utility computing:** a customer could upload tasks to a data center and be charged on a per-resource basis.
- **cloud computing** is characterized by an easily usable and accessible pool of *virtualized* resources.
  - **Infrastructure-as-a-Service** (**IaaS**) covering hardware and infrastructure layer
  - **Platform-as-a-Service** (**PaaS**) covering the platform layer
  - **Software-as-a-Service** (**SaaS**) in which their applications are covered

Types of Client-Server Architecture - Two Tier Architecture



- Clients connects to the data sources directly
- Logic resides on the client or the database
- Advantages:
  - **Ease in Developing Applications**: Applications can easily be developed due to simplicity.
  - **User Satisfaction**: Maximum user satisfaction is gained with accurate and fast prototyping of applications through robust tools.
  - **Applicable for Homogeneous Environment**: Since this contains static business rules it's more applicable for homogeneous environment.
  - **High Performance**: Database server and business logic is physically close, which offers higher performance.
- Limitations:
  - **Performance**: Performance begins to deteriorate as the population grows. this is due to the reason that each user has its own connection and the server has to keep all these connections live even when no work is being done.
  - **Security**: Each user must have their own individual access to the database, and be granted whatever rights may be required in order to run the application.
  - **Capability**: Independent of the type of client, much of the data processing has to be located in database making it totally dependent upon the capabilities and implementation provided by the database manufacturer. This limits the application functionality.
  - **Portability**: As the two-tier architecture is dependent upon the specific database implementation, porting an existing application to a different dbms becomes a major issue.

Types of Client-Server Architecture - Three Tier Architecture

- **Presentation Tier**: Occupies the top level and displays information related to service available on a website. This tier communicates with other tiers by sending results to the browser and other tiers in the network.
- **Application Tier**: Also called the middle tier, logic tier or business logic , this tier is pulled from the presentation tier. It controls application functionality by performing detailed processing.
- **Data Tier**: Houses database server where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic.