Cloud Computing

Pre-Requisites

Cloud Computing

- Reference Textbooks:

- Rajkumar Buyya, James Broberg,Andrzej M Goscinski, Cloud Computing: Principles and Paradigms, Wiley publication
- Toby Velte, Anthony Velte, Cloud Computing: A Practical Approach, McGraw-Hill Osborne Media
- Mastering Cloud Computing by Rajkumar Buyya, C. Vecchiola & S. Thamarai Selvi McGRAW Hill Publication
- Sosinsky B., “Cloud Computing Bible”, Wiley India

Course Outline

- Pre-requisites
- Introduction to Cloud Computing (CC)
- Cloud Architecture, Services and Applications
- Virtualization
- Management and Monitoring
- Cloud Security
- Cloud Tools

What Is A Distributed Computing?

- *Distributed computing is a field of computer science that studies distributed systems.*

- *Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one computer.*

-Wikipedia

What Is A Distributed System?

- *A distributed system is a collection of independent computers that appears to its users as a single coherent system.*

  - A. S. Tanenbaum et al.

- *A distributed system is defined as one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages.*

  - G. Coulouris et al.

- *A distributed system is one on which I cannot get any work done because some machine I have never heard of crashed.*

  - ***Leslie Lamport et al.***

Properties of Distributed Systems

- Why do we use DS?
  - Connecting Users and Resources
- Desired properties:
  - Transparency
  - Openness
  - Scalability
  - Enhanced Availability
  - Concurrency
  - independent failures
  - heterogeneity

Connecting users and resources

- Computer resources include all hardware resources (e.g., disks, printers), software resources (e.g., compiler) and data resources (e.g., files) etc.
- The main goal of a DS is to make it easy for users to access remote resources, and share them with other users in a controlled way.
- Connecting users and resources also makes it easier to collaborate and exchange information.
- Security becomes a big issue as connectivity and sharing increase.

Distribution Transparency

- Why Transparency?
  - To hide from the user and the application of the distribution of components, so that the system is perceived as a single system rather than a collection of independent components.
  - Achieving distribution transparency makes everyone into thinking that the collection of machines is simply an old-fashioned time-sharing system, instead of a collection of independent components.

Transparency in a Distributed System

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk

Different forms of transparency in a distributed system.

Openness

- **Openness** is the characteristic that determines whether the system can be extended in various ways.
- **Open distributed system:**

  Be able to interact with services from other open systems, irrespective of the underlying environment:

  - Systems should conform to well-defined interfaces
  - Systems should support portability of applications
  - Systems should easily interoperate

Openness

- **Achieving openness:**

  At least make the distributed system independent from heterogeneity of the underlying environment:
  - Hardware
  - Platforms
  - Languages

- In DSs, services are generally specified through interfaces, which are often described in an Interface Definition Language (IDL).

Scalability

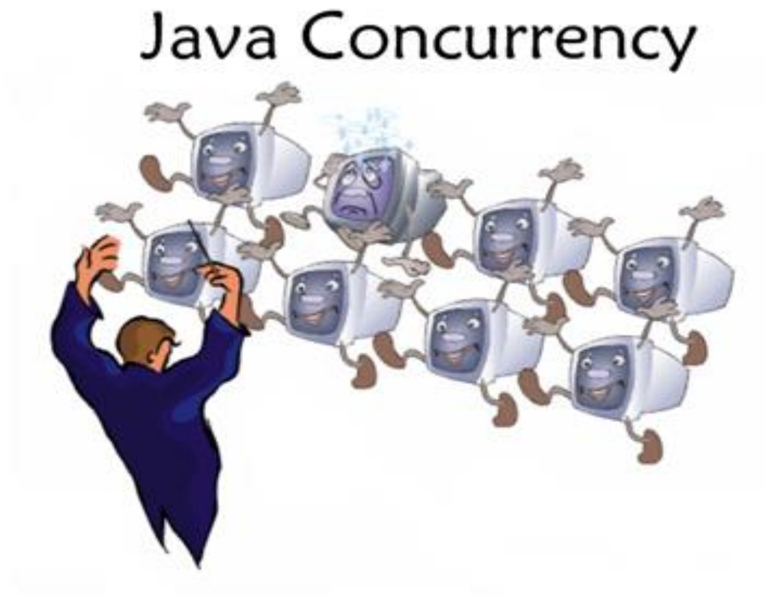
- The number of computers in the world is unlimited. Therefore, accessing shared resources should be nearly independent of the size of the network.
- A system is described as scalable if it will remain effective when there is a significant increase (or decrease) in the number of resources and the number of users.
- Scalability presents a number of challenges such as
  - how to control the cost of physical resources
  - and the performance loss,
  - and preventing software resources running out, etc.

Scalability

- Scalability of a system can be measured along three different dimensions:
  - A system is scalable in size if more users and resources can be easily added.
  - A system is geographically scalable if it allows users and resources to lie far apart.
  - A system is administratively scalable if it can be easy to manage even if it spans many independent administrative organizations.

Concurrency

- Provide and manage concurrent access to shared resources:
- Concurrency control by:
  - Fair scheduling
  - Preserve dependencies (e.g. distributed transactions)
  - Avoid deadlocks

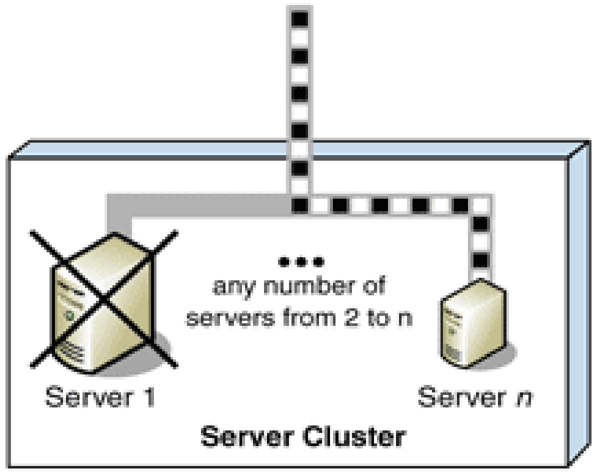


Heterogeneity

- Heterogeneous components must be able to interoperate across different:
  - Operating systems
  - Hardware architectures
  - Communication architectures
  - Programming languages
  - Software interfaces
  - Security measures
  - Information representation

Fault Tolerance

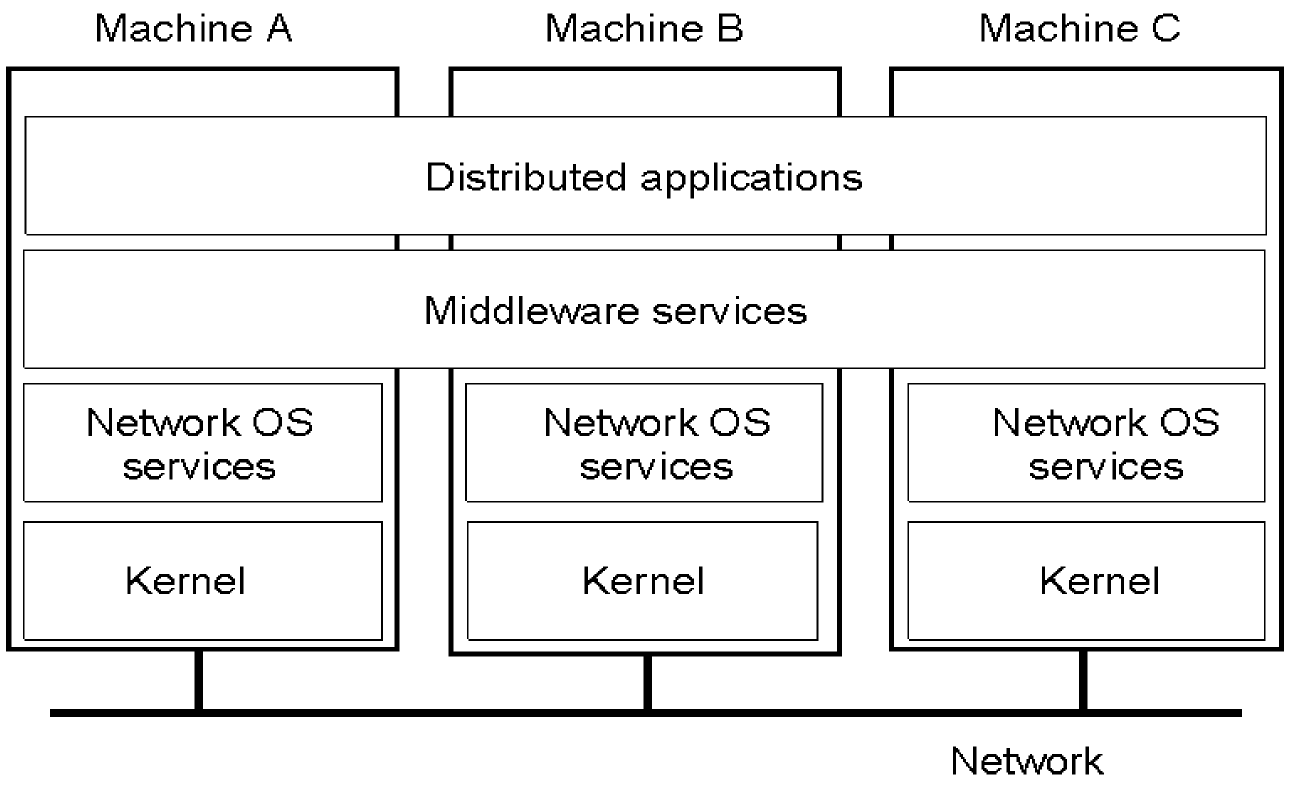
- **Failure:** an offered service no longer complies with its specification
- **Fault:** cause of a failure (e.g. crash of a component)
- **Fault tolerance**: no failure despite faults



Middleware

- “a software layer between applications and the OS whose purpose is to mask heterogeneity of the underlying components and to provide a convenient programming model to application programmers.”
- Middleware is a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems.
- It is defined as a layer of software above the operating system but below the application program that provides a common programming abstraction across a distributed system.

Positioning Middleware



- General structure of a distributed system as middleware.

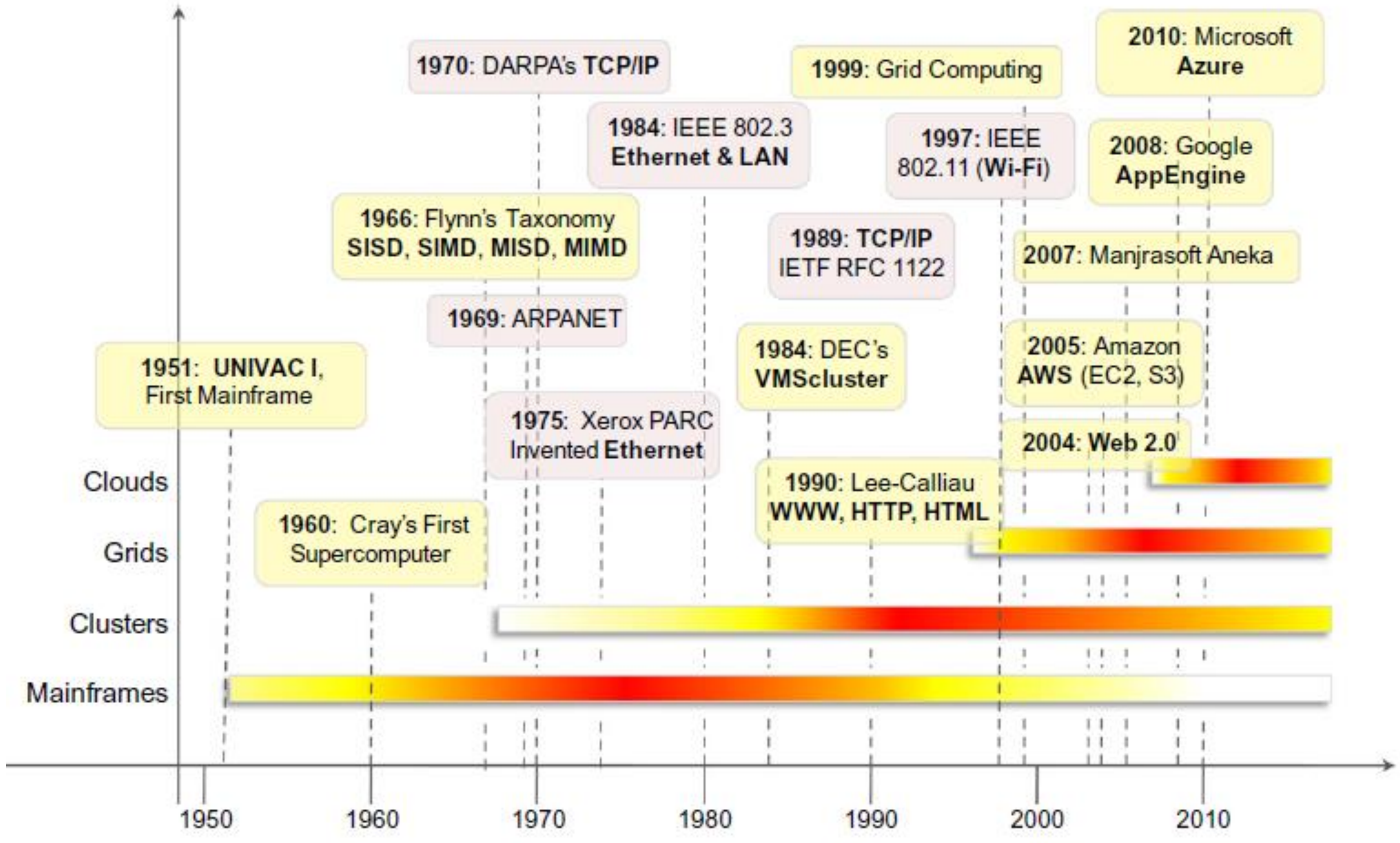
Middleware

- Middleware frameworks are designed to mask some of the kinds of heterogeneity that programmers of distributed systems must deal with.
- They always mask heterogeneity of networks and hardware. Most middleware frameworks also mask heterogeneity of operating systems or programming languages, or both.
- A few such as CORBA also mask heterogeneity among vendor implementations of the same middleware standard.

Examples of Middleware

- Database access technology - e.g ODBC (Open DataBase Connectors)
- Java’s database connectivity API : JDBC
- Remote computation products - e.g RPC (Remote Procedure Call) and RMI (Java Remote Method Invocation)
- Distributed Computing Environment (DCE) products, Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM)

The evolution of distributed computing technologies, 1950s–2010s.

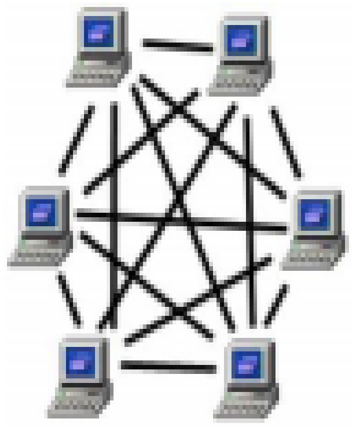

1970: DARPA's TCP/IP
1999: Grid Computing
2010: Microsoft Azure
1984: IEEE 802.3 Ethernet & LAN
1997: IEEE 802.11 (Wi-Fi)
2008: Google AppEngine
1966: Flynn's Taxonomy SISD, SIMD, MISD, MIMD
1989: TCP/IP IETF RFC 1122
2007: Manjrasoft Aneka
1969: ARPANET
1951: UNIVAC I, First Mainframe
1984: DEC's VMScluster
2005: Amazon AWS (EC2, S3)
1975: Xerox PARC Invented Ethernet
2004: Web 2.0
Clouds
1990: Lee-Calliau WWW, HTTP, HTML
1960: Cray's First Supercomputer
Grids
Clusters
Mainframes
1950
1960
1970
1980
1990
2000
2010

Classification of Distributed Computing

1) Peer-to-peer Computing
2) Cluster Computing
3) Utility Computing
    1) Grid Computing
    2) Cloud Computing

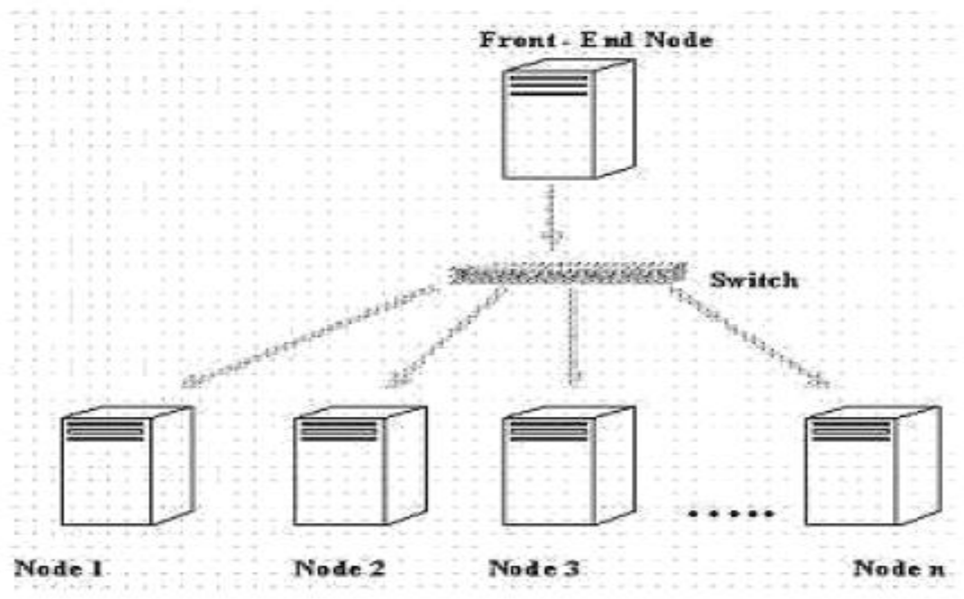
Peer to Peer Computing

- In a P2P system, every node acts as both a client and a server, providing part of the system resources.
- No central coordination or no central database is needed.



Cluster Computing

- **Cluster** - *"A type of parallel or distributed processing system, which consists of a collection of interconnected stand-alone computers cooperatively working together as a single, integrated computing resource"* [Buyya].



Cluster Computing

- The components of a cluster are connected to each other through fast local area networks
- Cluster computing provides high computation by employing parallel programming, which is use of many processors simultaneously for a number of or a single problem.
- Another reason is fault tolerance which is actually the ability of a system to operate gracefully even in the presence of any fault.

Utility Oriented Computing

- **Utility computing**, is a service provisioning model in which a service provider makes computing resources and infrastructure management available to the customer as needed, and charges them for specific usage rather than a flat rate.

Utility Oriented Computing

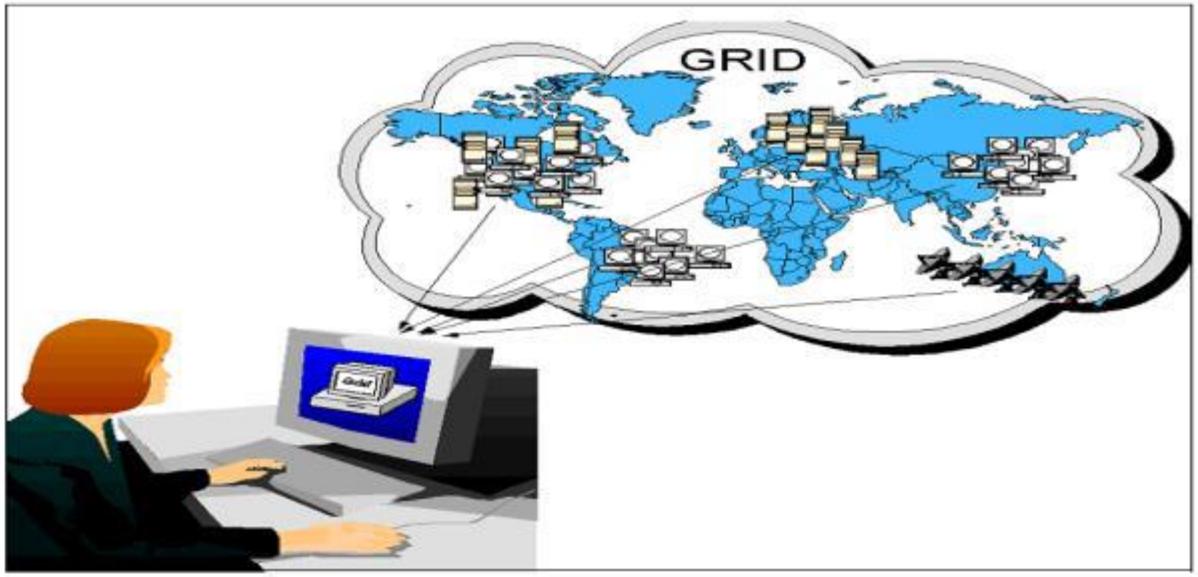
- *utility* is used to make an analogy to other services, such as electrical power, that charge for the resources based on usage rather than on a flat-rate basis.
- This approach, sometimes known as *pay-per-use* or metered services, is sometimes used for the consumer market as well, for Internet service, Web site access, file sharing, and other applications

Grid Computing

- **Grid -** *"A type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements"* [Buyya].
- Here individual user gets access to the resources (like processors, storage, data etc.) on demand with little or no knowledge of the fact that where those resources are physically located.

Grid Computing

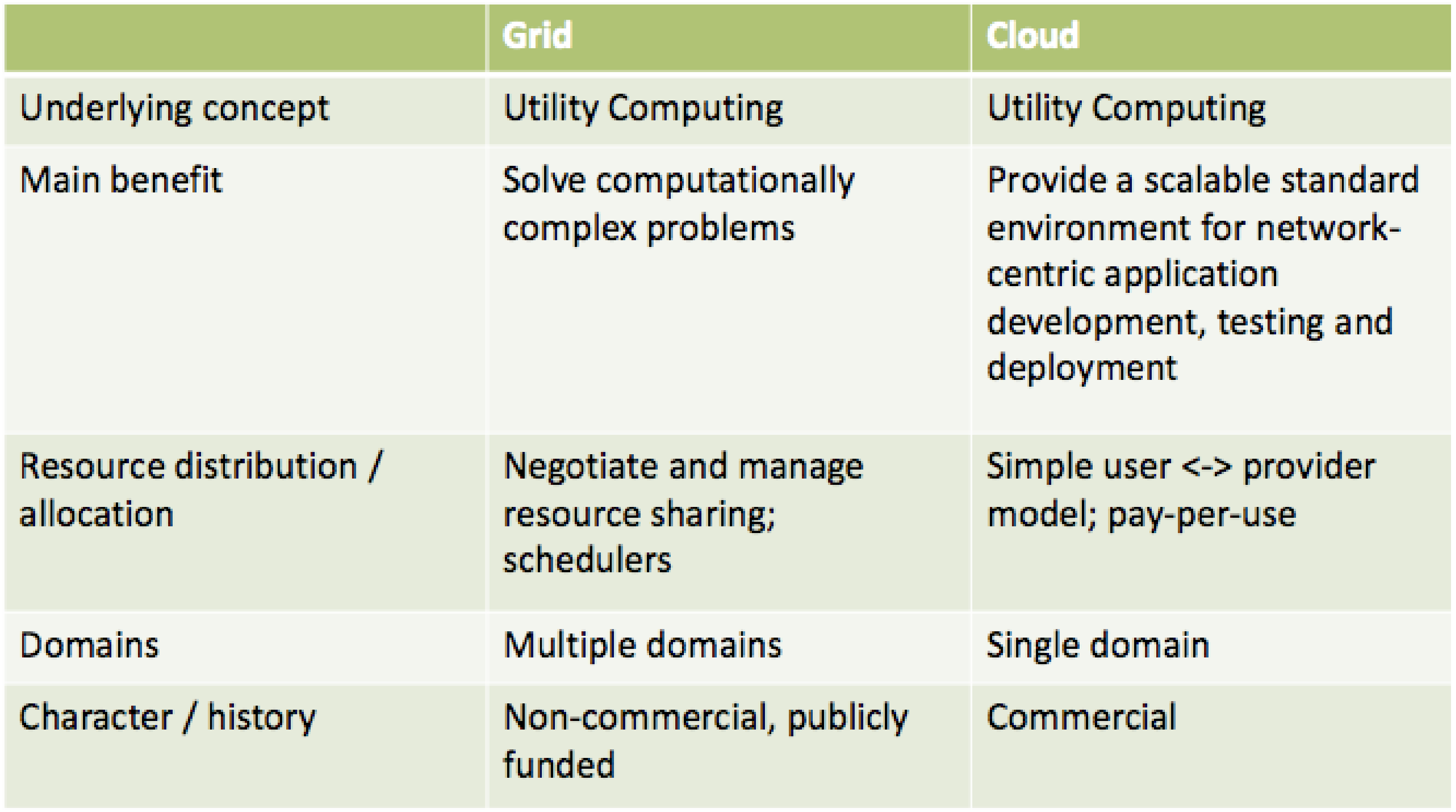
- It employs use of multiple clusters that are loosely coupled, heterogeneous and are geographically dispersed.



Grid v/s Cluster Computing

Grid Computing	Cluster Computing
Heterogeneous	Homogenous
The computers that are part of a grid can run different hardware as well as have different OS	The cluster computers all have the same hardware and OS
Grid can make use of spare computing power on a desktop computer	The machines in a cluster are dedicated to work as a single unit as well as nothing else
Grid are inherently distributed by its nature over a LAN, MAN or WAN	The computers in the cluster are normally contained in a single location
Every node is autonomous (it has its own resource manager as well as behaves like an independent entity)	Whole system (all nodes) behave like a single system view as well as resources are managed by centralized resource manager

Grid v/s Cloud Computing

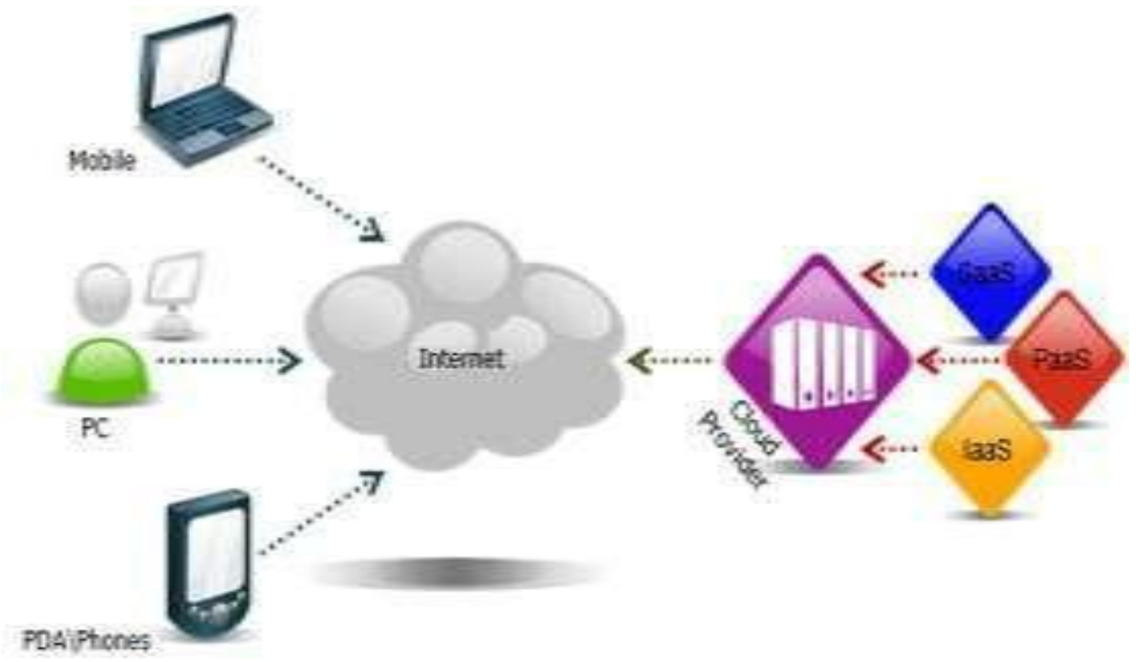


	Grid	Cloud
Underlying concept	Utility Computing	Utility Computing
Main benefit	Solve computationally complex problems	Provide a scalable standard environment for network-centric application development, testing and deployment
Resource distribution / allocation	Negotiate and manage resource sharing; schedulers	Simple user <-> provider model; pay-per-use
Domains	Multiple domains	Single domain
Character / history	Non-commercial, publicly funded	Commercial

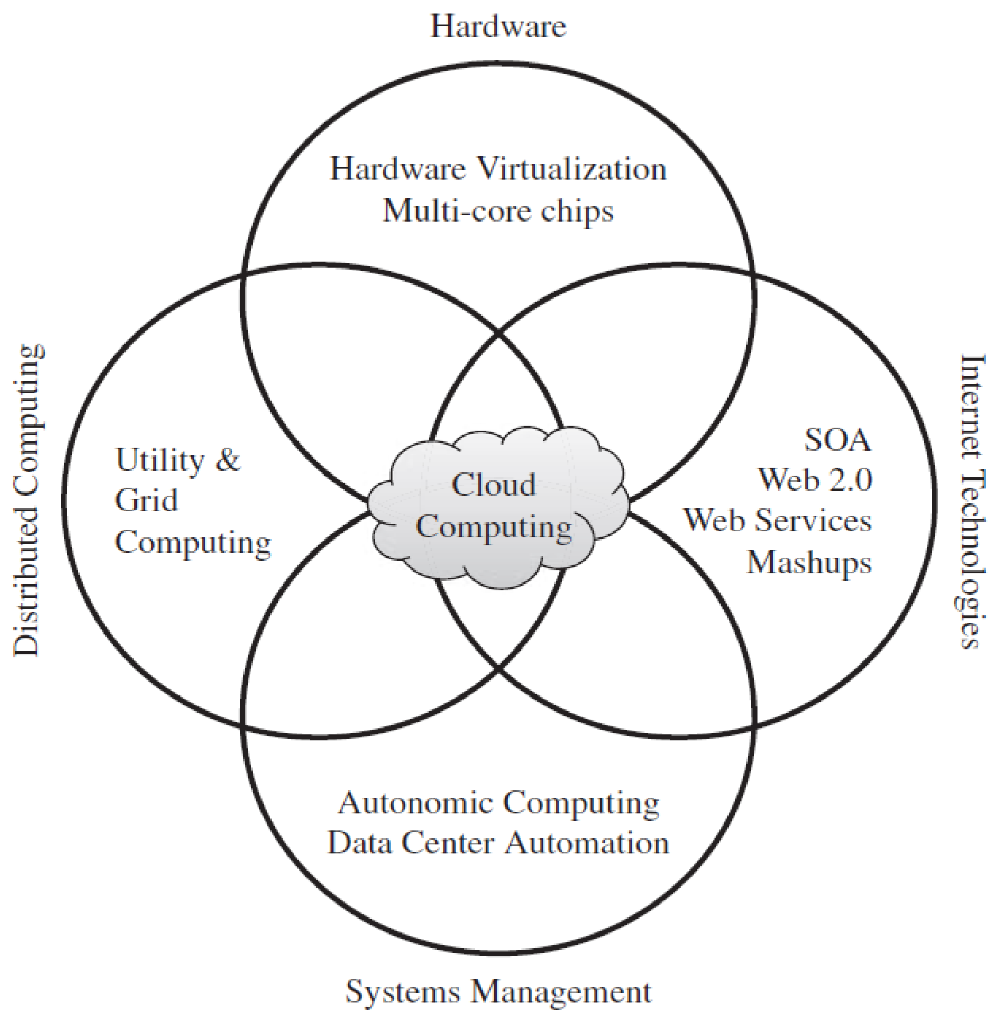
Cloud Computing

- **Cloud –** *“A type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements” [Buyya].*
- The main principle behind cloud computing model is to offer computing, storage, and software as a service or as a utility.
- We just need internet to use these utilities.

Cloud Computing



Cloud Computing



Service-oriented computing

- Service orientation is the core reference model for cloud computing systems.
- This approach adopts the concept of services as the main building blocks of application and system development.
- Service-oriented computing(SOC) supports the development of rapid, low-cost, flexible, interopera-ble, and evolvable applications and systems.

Service-oriented computing

- **Service-Oriented Computing (SOC)** is the computing paradigm that utilizes services as fundamental elements for developing applications/solutions.
- **Services** are self- describing, platform-agnostic computational elements that support rapid, low-cost composition of distributed applications.
- **Services** perform functions, which can be anything from simple requests to complicated business processes.