Chapter One

Distributed Systems

Collected from different books, Journals, and Sites

Prepared By Prof. Fatma Omara

Computer Science Department

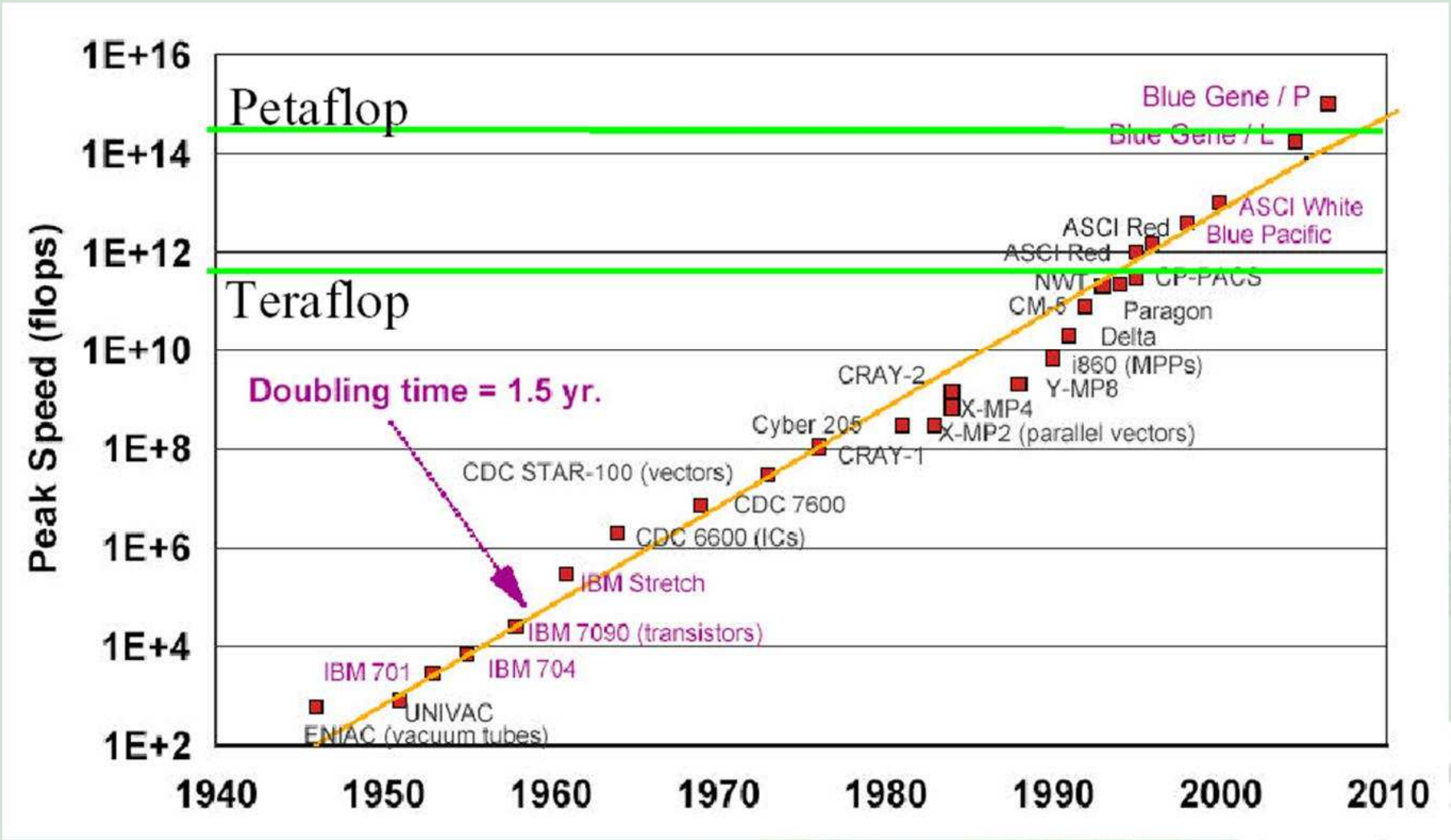
Faculty of Computers & Artificial Intelligence

2022-2023

Introduction To Distributed Systems

- What is a Distributed system?
- Why do we need distributed Systems?
- History of Distributed Systems.
- Applications of Distributed Systems.
- Goals and Objectives.

Computer processor performance evolution:

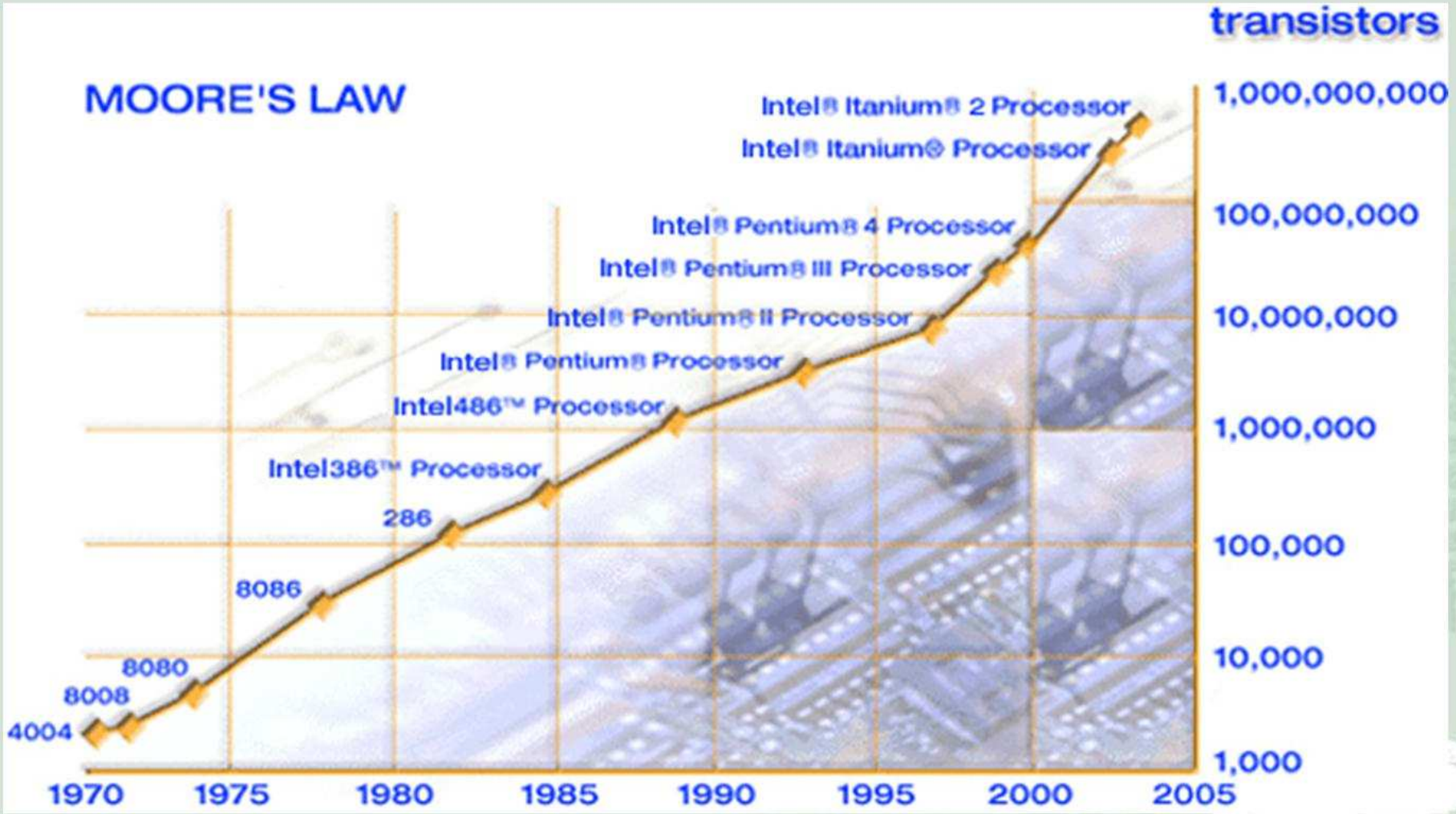


Moore's Law

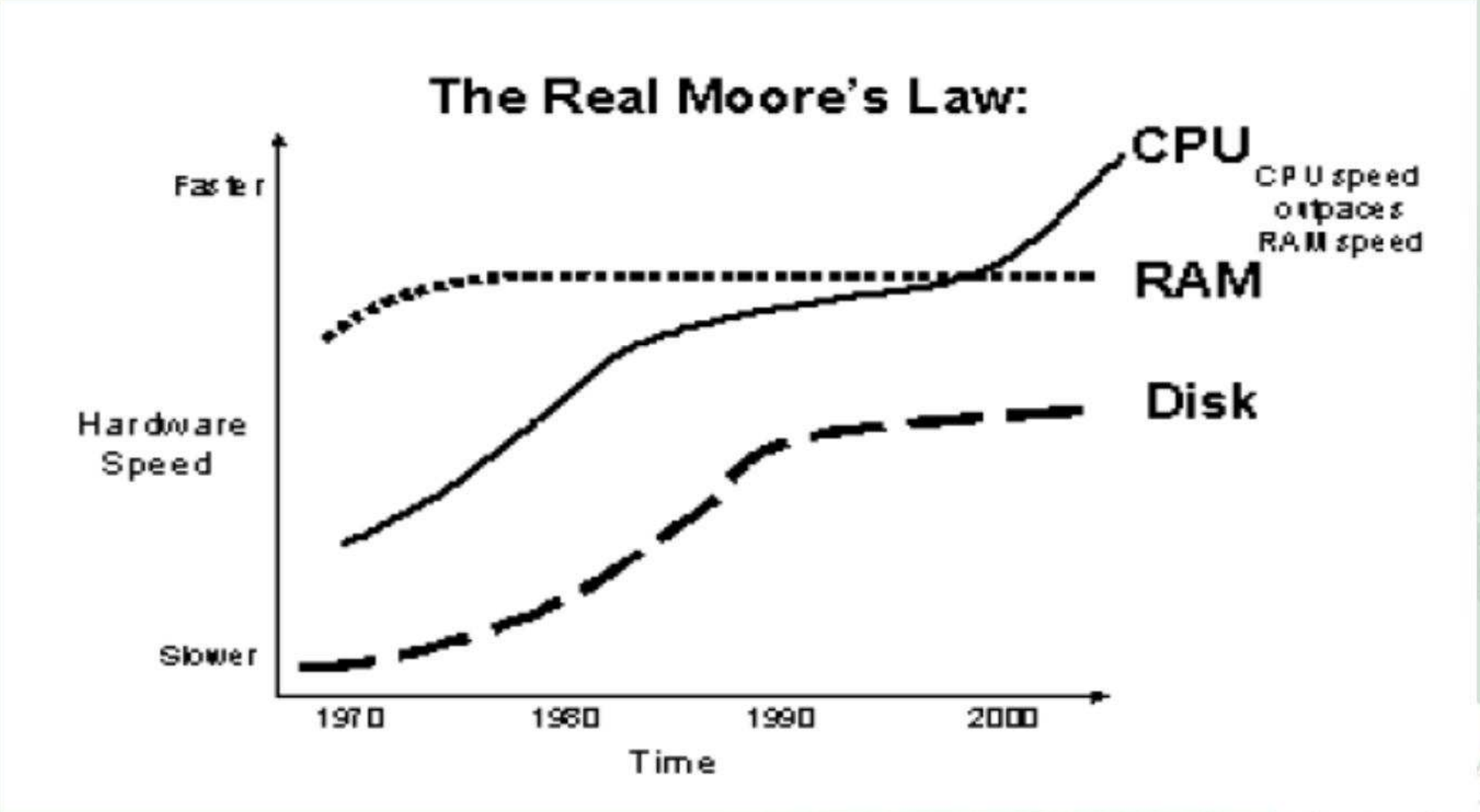
- Moore's observation in 1965:
  - number of transistors per square inch on integrated circuits had doubled ***every year***

- Moore's revised observation in 1975:
  - the space slowed down a bit, but data density had doubled approximately every ***18 months***

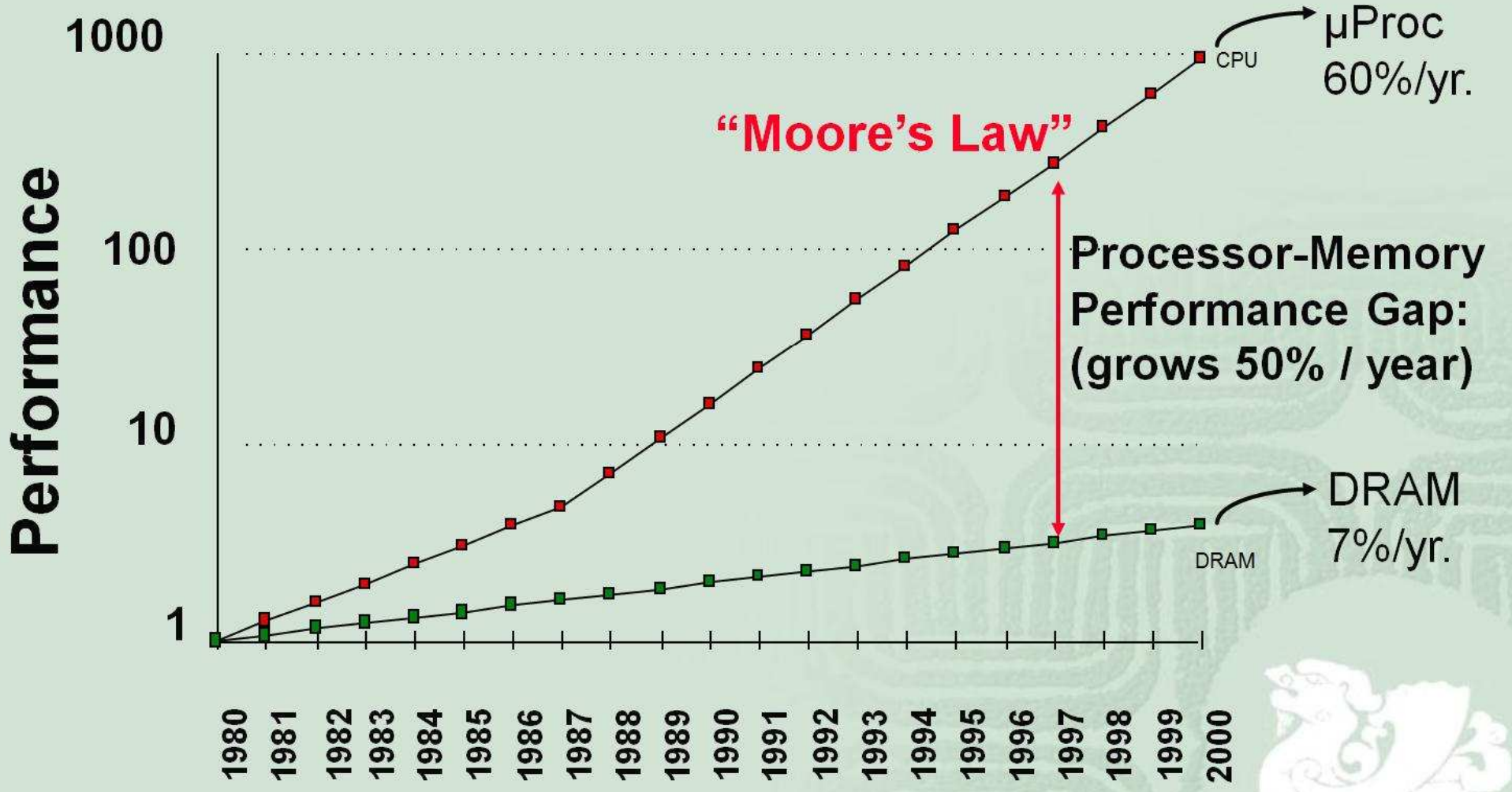
Gordon Moore(1965) Prediction:



CPU, Memory, and Disk Speed



Processor-Memory Performance Gap



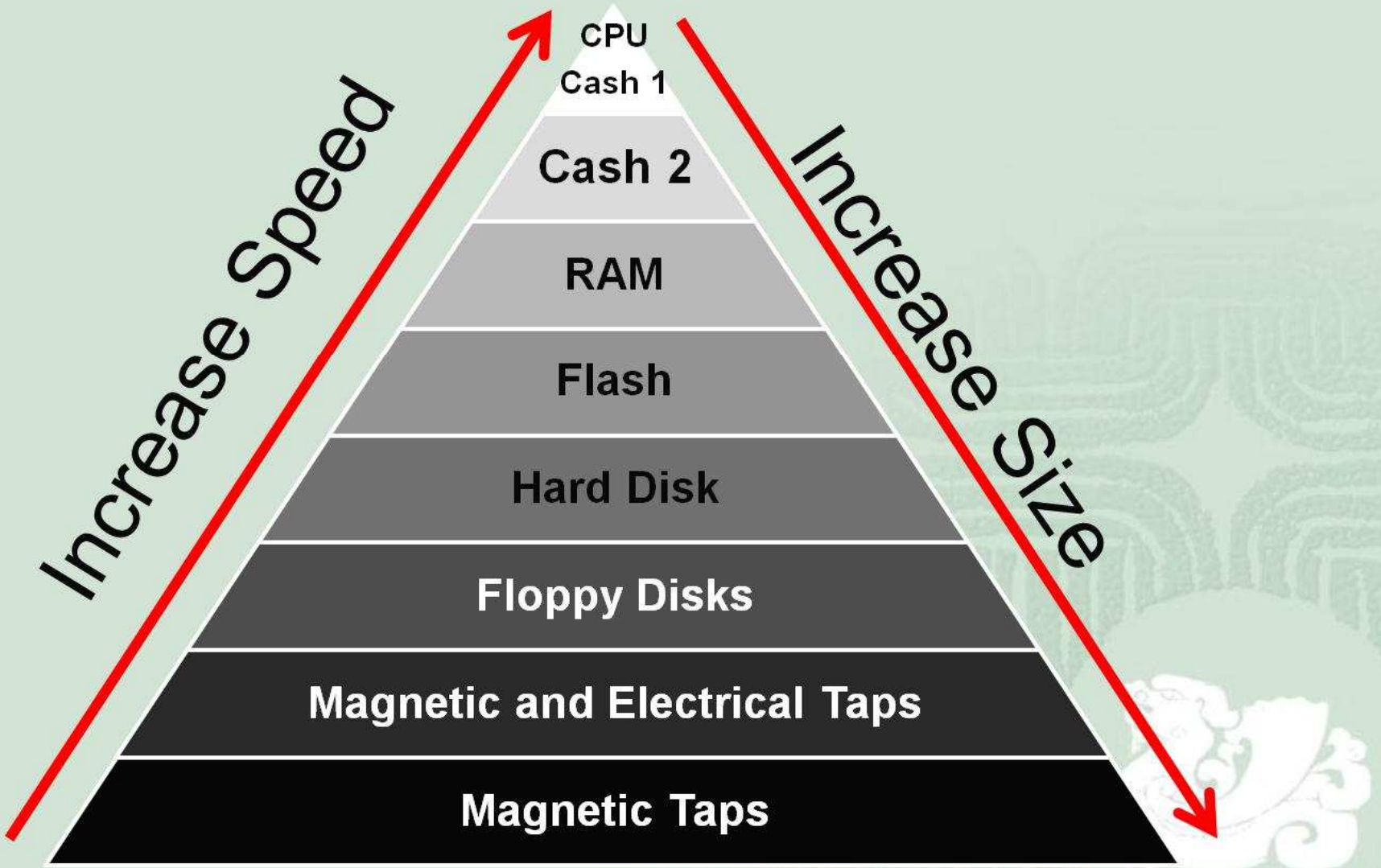
Possible Solutions

- A hierarchy of successively fast memory devices (multilevel caches)
- Location of data reference (data Locality)
- Efficient programming can be an issue
- Parallel systems may provide
  1. larger aggregate cache
  2. higher aggregate bandwidth to the memory system

By Hierarchy of different types of memories

Hierarchy of Successful Fast Storage

To guaranty data locality



Technology Development

Three Technology Advances:

- Development of powerful microprocessors
- Development of high-speed networks
- Development of denser and cheaper memory/storage

Easy:

put together large # of powerful processors connected by a high speed network.

Hard:

SOFTWARE! SOFTWARE! SOFTWARE!

Definition of a Distributed System

- A collection of ***independent computers*** that appears to its users as a ***single coherent system***. (Tanenbaum)
- A collection of (perhaps) ***heterogeneous nodes*** connected by one or more ***interconnection networks*** which provides access to system-wide ***shared resources*** and ***services***.
- A system in which ***hardware*** or ***software*** components located at ***networked computers communicate*** and ***coordinate*** their actions only by "***message passing***".
- A system that consists of a collection of ***two or more independent computers*** which ***coordinate their processing*** through the ***exchange of synchronous or asynchronous message passing***.
- A collection of ***autonomous computers*** linked by a ***network*** with **software designed to produce an integrated computing facility**.

Characteristics of Distributed Systems

- **Multiple Computers:**

More than one physical computer, each consisting of ***CPUs***, ***local memory***, and possibly stable ***storage***, and ***I/O paths*** to connect it with the environment.

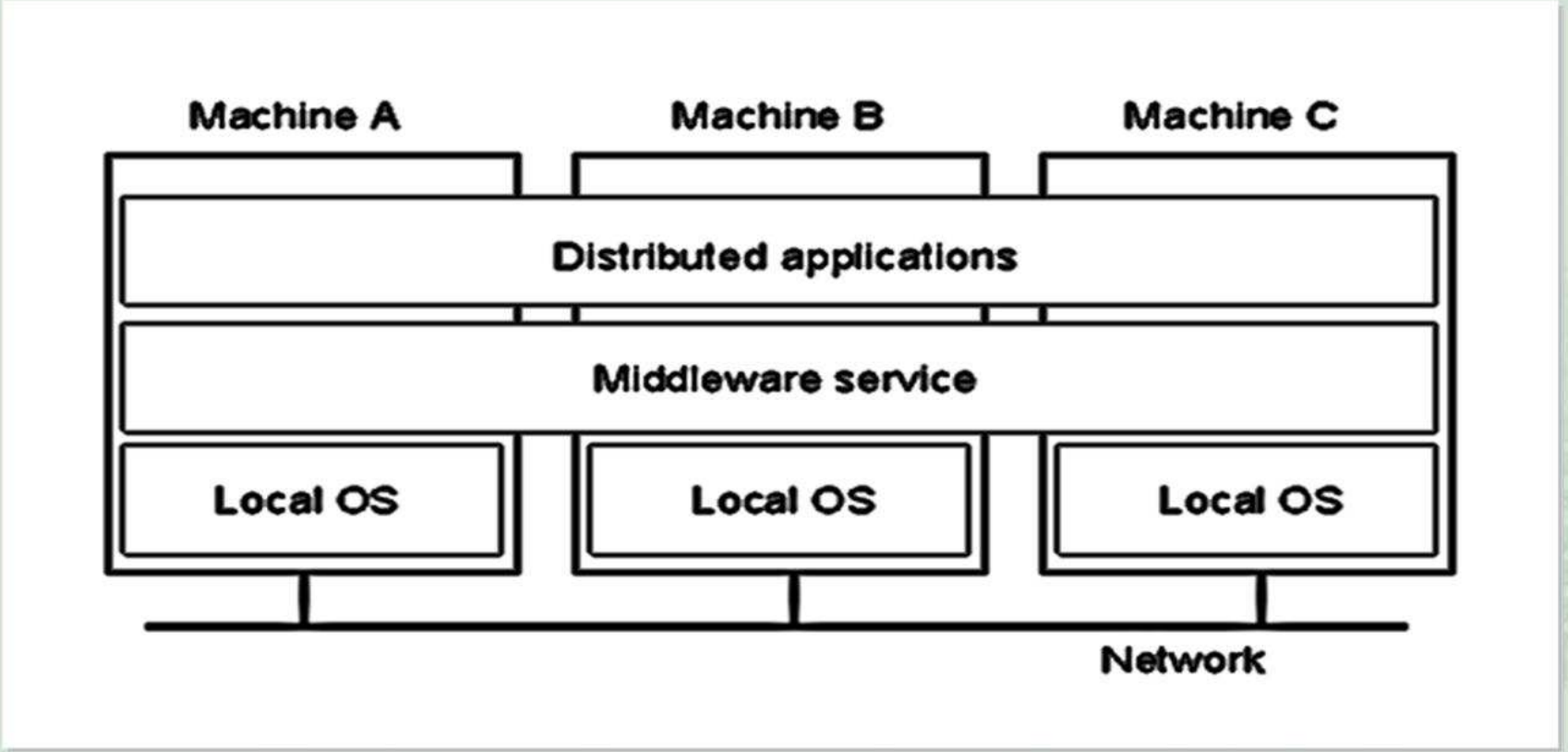
- **Interconnections:**

Mechanisms for communicating with other nodes via a network.

- **Shared State:**

If a subset of nodes cooperate to provide a service, a shared state is maintained by these nodes. The shared state is ***distributed*** or ***replicated*** among the participants.

An Abstract View Distributed Systems



A distributed system organized as middleware.

Note that

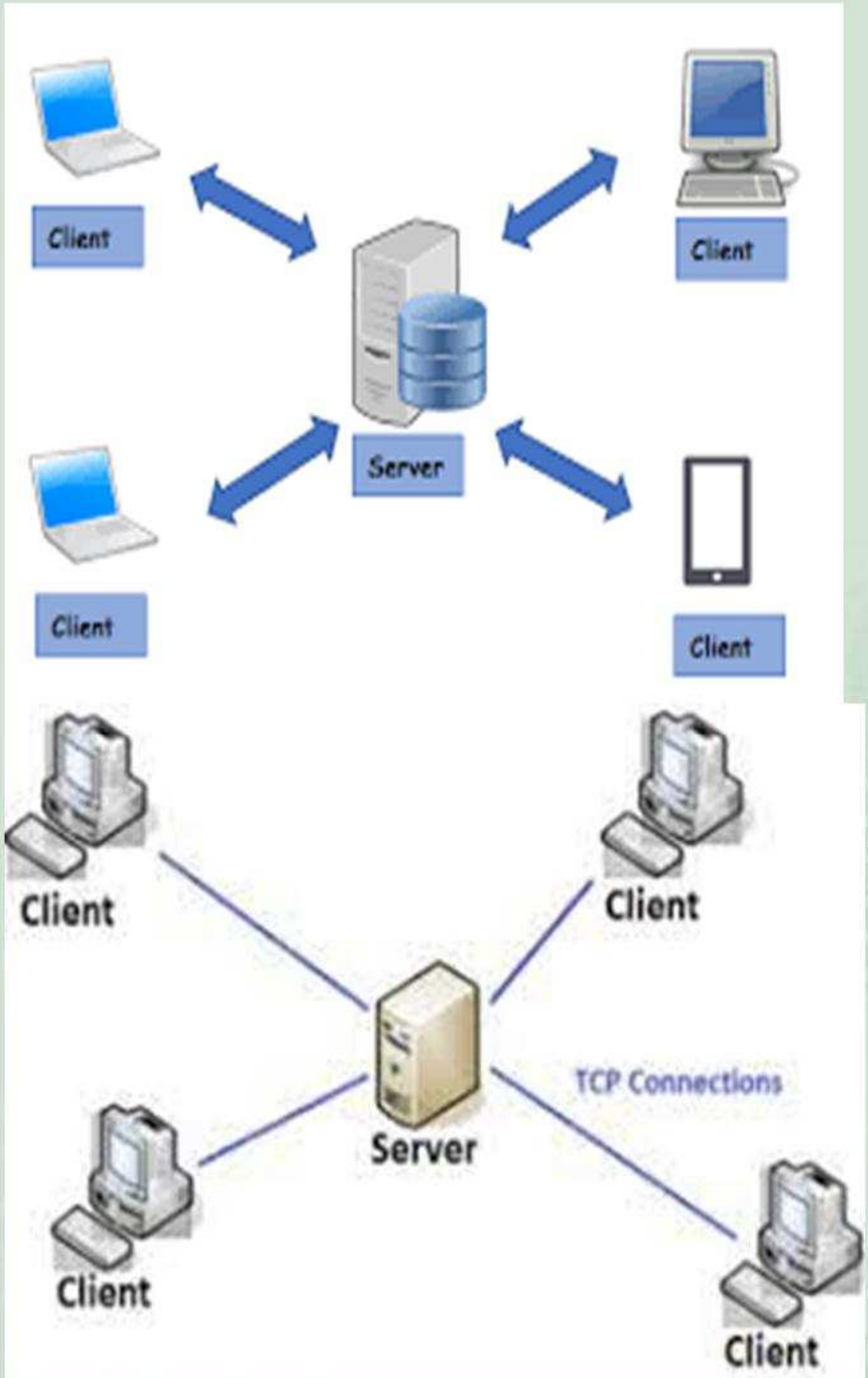
Middleware layer extends over multiple machines.

Centralized Systems

- System shared by users all the time
- All resources accessible
- Software runs in a single process
- Single physical location
- Single point of control
- Single point of failure

Example:

Airplane booked, Banks



Decentralised Systems

- Multiple autonomous components
- Components shared by users
- Some resources may not be accessible
- Software can run in concurrent processes on different processors
- Multiple physical locations
- Multiple points of control
- Multiple points of failure
- No global time
- No shared memory (in most cases)

Example: Gird, Cloud

Distributed System Reasons

- **Functional Distribution**:
  Computers have different functional Capabilities:
  - Client / server
  - Host / terminal
  - Data gathering / data processing
- **Sharing of Resources**:
  - With specific functionalities (***Example: P2P***)
- **Inherent Distribution**:
  Stemming from the application domain, e.g.
  - Cash register and inventory systems for supermarket chains
  - Computer supported collaborative work
- **Load Distribution / Balancing**:
  assign tasks to processors such that the overall system performance is optimized (HPC).

Distributing Systems Reasons-Cont…

- **Replication of Processing Power:**
  - Independent processors working on the same task
  - Distributed systems consisting of collections of microcomputers may have processing powers that no supercomputer will ever Achieve
  - 10000 CPUs, each running at 50 MIPS, yields 500000 MIPS, then
    - Instruction to be executed in 0.002 nsec
    - Equivalent to light distance of 0.6 mm
    - Any processor chip of that size would melt immediately

- **Physical Separation:**

  Systems that rely on the fact that computers are physically separated (e.g., to satisfy ***reliability*** requirements).

- **Economics:**

Collections of microprocessors offer a better price/performance ration than large mainframes.

–mainframes: 10 times faster, 1000 times as expensive

Applications

Computing Dominated Problems (Distributed Computing): [HPC]

Mathematical Computations, Environmental and Biological Modeling, Economic and Financial modeling, Graphics rendering for visualization, Network Simulations.

Storage Dominated Problems (Distributed Data):

Data Mining, Image Processing, Information retrieval, Insurance Analysis, IOT applications

Communications Dominated Problems (Network Computing):

Transaction processing, Video on Demand, E-com, Electronic banking, electronic shopping

Common Distributed Computing Examples

- Network file system, network printer etc..
- ATM (cash machine)
- Distributed databases
- Network computing
- Global positioning systems (GPS)
- Retail point-of-sale terminals
- Air-traffic control
- Enterprise computing
- WWW

Basics of Distributed Systems:

- Networked computers (***Tightly*** or ***loosely*** coupled) that provide a degree of operation transparency

Distributed Computer System = independent processors + networking infrastructure

- Communication between processes (on the same or different computer) using ***message passing*** technologies is the basis of distributed computing

Relation between tasks, Job, processes, processors?!

Goals of Distributed Systems

[1] Resource Sharing

- Easy for users to access remote resources.
- Resources types
  - Hardware, e.g. printer, scanner, camera
  - Data, e.g. file, database, web page
  - More specific functionality, e.g. search engine, file

Some Definitions:

- ***Service***
  - manage a collection of related resources and present their functionalities to users and applications. the requesting process called ***Client***
- ***Server***
  - a process on networked computer that accepts requests from ***Client*** processes on other computers to perform a ***service*** and responds appropriately
- ***Remote invocation***
  - A complete interaction between client and server, from the point when the ***client*** sends its request to when it receives the ***server***'s response

Goals of Distributed Systems- Cont..

[2] Transparency

- How to achieve "single-system image"? How to hide distribution from users or programs?
- Is it a good idea?
  - Distribution transparency is a nice a goal, but achieving it is a different story.
    - Sometimes requires trade off ***transparency*** for ***performance***

Goals of Distributed Systems – Cont..

[2] Transparency:

- Hide that processes and resources are physically distributed across multiple computers.
- Transparency has different aspects:

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 (i.e., Cold Migration)
Relocation	Hide that a resource may be moved to another location while in use (i.e., Hot Migration)
Replication	Hide that a resource may be redundant and shared by several competitive users
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

Transparency - Cont..

- **Access Transparency**
  - Using identical operations to access local and remote resources, e.g. a Graphical User Interface (GUI) with folders
- **Location Transparency**
  - Resources to be accessed without knowledge of their location, e.g. URL

Availability!!

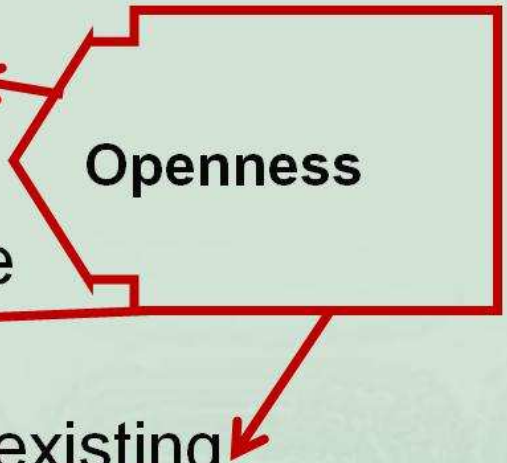
- **Concurrency Transparency**
  - Several processes operate concurrently using shared resources without interference with between them
- **Replication Transparency**
  - Multiple instances of resources to be used to increase ***reliability*** and ***performance*** without knowledge of the replicas by users or application programmers,

Transparency - Cont..

- **Failure Transparency**
  - Users and applications complete their tasks despite the failure of hardware and software components (Allow fail and recovery) (***reliability***)
    - e.g., email
- **Mobility Transparency**
  - Movement of resources and clients within a system without affecting the operation of users and programs (***Migration*** and ***Relocation***),
    - e.g., mobile phone
- **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

Goals of Distributed Systems – Cont..

[3] Openness:

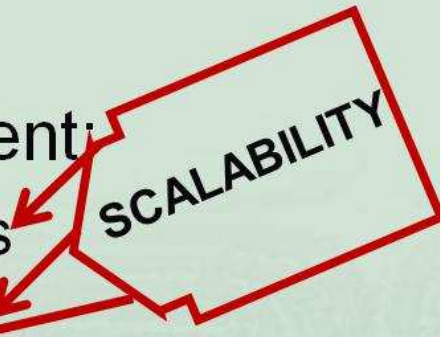


- Openness is concerned with extensions and improvements of distributed systems.
- Detailed interfaces of components need to be published.
- New components have to be integrated with existing components.
- It is determined by ***the degree to which new resource can be added and made available for using by a variety of client programs***.
  - e.g. Web, and Internet
- Differences in data representation of interface types on different processors (of different vendors) have to be resolved. (***Heterogeneity***)

Openness

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



Achieving Openness

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

Goals of Distributed Systems – Cont..

[4] Scalability:

- easy to expand and manage.
- A system is described as scalable
  - if will remain effective when there is a significant increase in the ***number of resources*** **and the** ***number of users.*** e.g., Internet

Date	*Computers*	*Web servers*	*Percentage*
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12

Scalability

- The challenge is to build distributed systems that scale with the increase in the number of CPUs, users, and processes, larger databases, etc.
- Scalability along several dimensions:
  - Number of users and/or processes (size scalability)
  - Maximum distance between nodes (geography scalability)
  - Number of administrative domains (administrative scalability)

➢ ***Design components to be scalable!***

Scalable System?!

User Requirements

- What services the system can provide?
- How easy to use and manage the system?
- What benefits the system can offer?
- What is the ratio of performance/cost?
- How reliable the system is?
- How secure the system can guarantee?

Reliability!!!

Security!!!

Distributed Computer System Metrics

- **Latency** – network delay before any data is sent
- **Bandwidth** – maximum channel capacity (analogue communication Hz, digital communication bps)
- **Granularity** – relative size of units of processing required. Distributed systems operate best with ***coarse grain*** granularity because of the slow communication compared to processing speed in general
- **Processor speed** – MIPS, FLOPS
- **Reliability** – ability to continue operating correctly for a given time
- **Fault tolerance** – flexibility to partial system failure
- **Security** – policy to deal with threats to the communication or processing of data in a system
- **Administrative/management domains** – issues concerning the ownership and access to distributed systems components

Performance

Various performance metrics:

- Response time
- Throughput
- System utilization
- Network capacity utilization
- Key issue in parallelizing computations in a distributed system?

overhead of message communication

Distributed Programming Paradigms

- Client/server model
- Remote procedure calls
- Distributed File Systems
- Group communication and multicasts
- Distributed transactions
- Distributed shared memory
- Distributed object-based systems
- Publish-subscribe model
- Peer-to-peer model
- The Web

Distributed Systems Challenges (1)

1. Absence of a global clock

- Due to ***asynchronous message passing*** there are ***limits on the precision*** with which processes in a distributed system can synchronize their clocks.

2. Absence of a global state

- In the general case, there is no single process in the distributed system that would have a knowledge of the current global state of the system
  - Due to ***concurrency*** and ***message passing communication***

3. Specific failure modes

- Processes run autonomously, in isolation
  - Failures of individual processes may remain undetected.
  - Individual processes may be unaware of failures in the system context

Distributed Systems Challenges (2)- Cont..

4. Heterogeneity

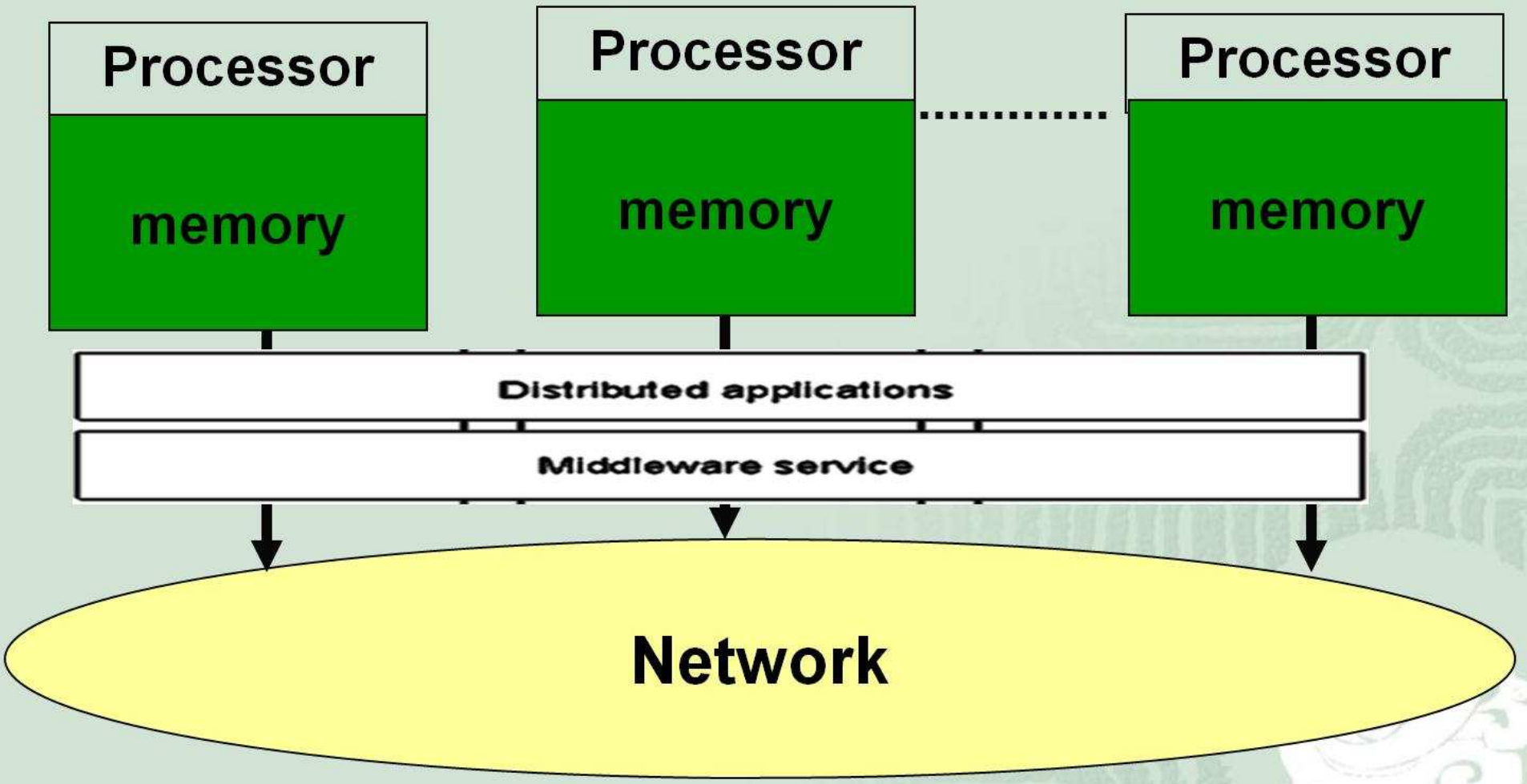
Distributed systems developed to many different kinds of software and hardware.

Heterogeneity at many levels:

- **Network**: different kinds of software and hardware
- **Operating system**: different APIs to internet
- **Programming languages**: many different Programming Languages
- **Data**: different representations of data (Big Indian, Small Indian)
- **Hardware**: Different clock cycles and memory capacity
- **Data structures**: Implementations by different developer

Distributed Systems Challenges (2)

Solution of Heterogeneity



Distributed Systems Challenges (2)- Cont..

Heterogeneity (Solution)

- **Middleware layer**: (e.g., **C**ommon **O**bject **R**equest **B**roker **A**rchitecture ((CORBA))
  - integrates many computing devices to act as a ***coordinated computational resource*** and ***hide different topologies communication networks and computing devices***
- **Services that can be regarded as middleware**:
  - **MOM**: **M**essage **O**riented **M**iddleware
  - **ORBs**: **O**bject **R**equest **B**rokers
  - **ESB**: **E**nterprise **S**ervice **B**us
  - **Uniform high level API**.

Distributed Systems Challenges (3)

5. Openness

– Ensure extensibility and maintainability of systems

- Adherence to standard interfaces

6. Security

- Privacy, Authentication, Availability, Trusting, Authorization

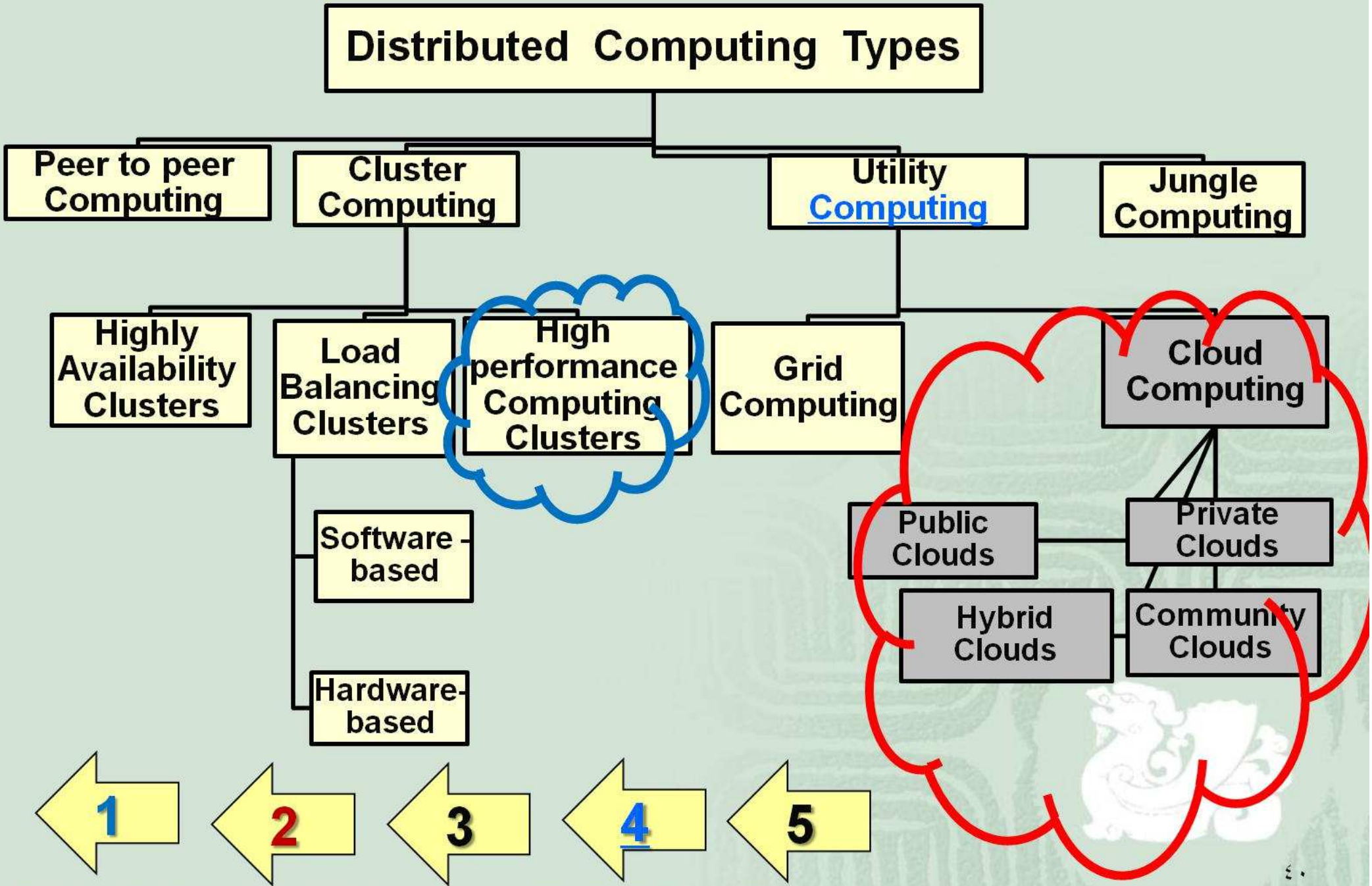
7. Concurrency

- Consistent *scheduling of concurrent processes* so that dependencies (***Proceeding***) are preserved, (e.g., in ***concurrent transactions***)
- Allow several processes to operate concurrently using shared resources in a consistent fashion
- Avoidance of ***dead lock*** and ***life lock*** problems

Modeling a Distributed System

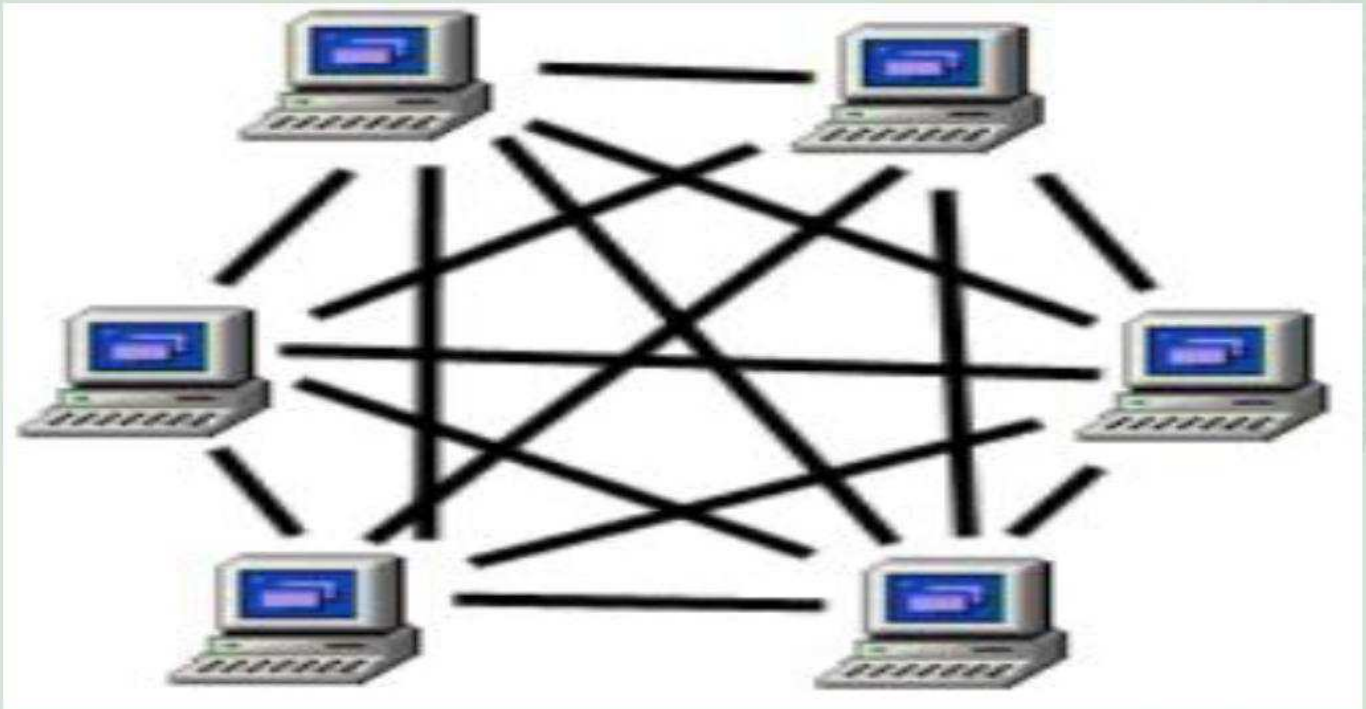
- **Asynchronous System**
  - No bound on time to deliver a message
  - No bound on time to compute
    - Internet essentially asynchronous
- **Synchronous System**
  - Known bound on time to deliver a message
  - Known bound on time to compute
    - LAN/cluster essentially synchronous
- **Partially Synchronous System**
  - Initially system is asynchronous
  - Eventually the system becomes synchronous

Communication Protocols


Distributed Computing Types
Peer to peer Computing
Cluster Computing
Utility Computing
Jungle Computing
Highly Availability Clusters
Load Balancing Clusters
High performance Computing Clusters
Grid Computing
Cloud Computing
Public Clouds
Private Clouds
Hybrid Clouds
Community Clouds
Software -based
Hardware-based
1
2
3
4
5

Peer to Peer Computing (P2P)

- Large number of distributed resources connect by a network
- Every node acts as both a *client* and a *server*
  - No master-slave relationship exists among the peers.
- According to P2P system, **no peer machine has a global information of the entire P2P system**
  - No *central* coordination or no *central* database



Absence of a global state

Peer to Peer Computing (P2P)

- Peer-to-peer computing is the sharing of computer resources and services by direct exchange between systems.
- **These resources and services include:**
  - Information Exchange
  - Processing Cycles,
  - Cache Storage, and
  - Disk Storage for Files.
- **What is driving P2P?**
  - Reduced The Load On Servers
  - Inexpensive Computing Power
  - Bandwidth and Storage

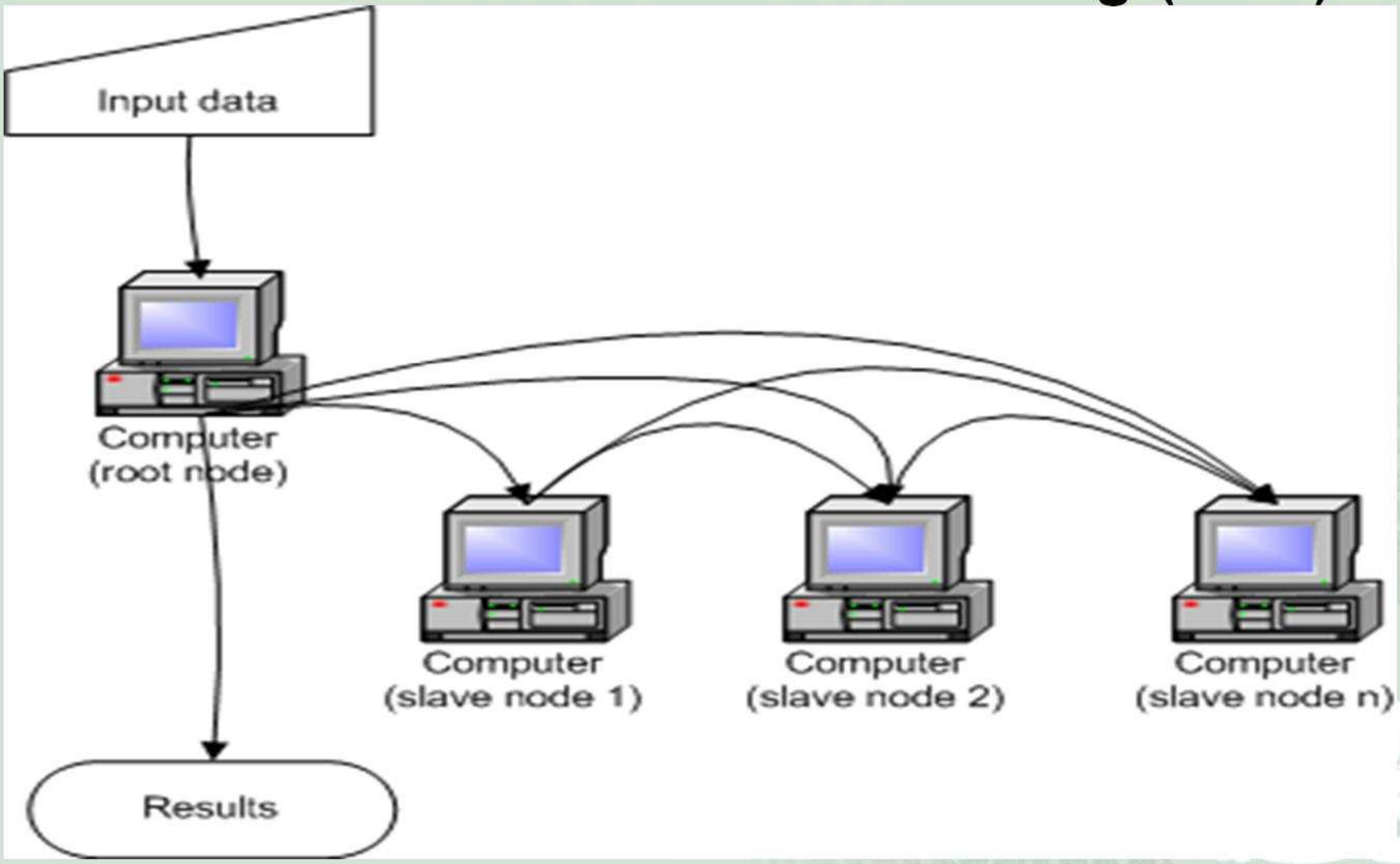
Back

Cluster Computing

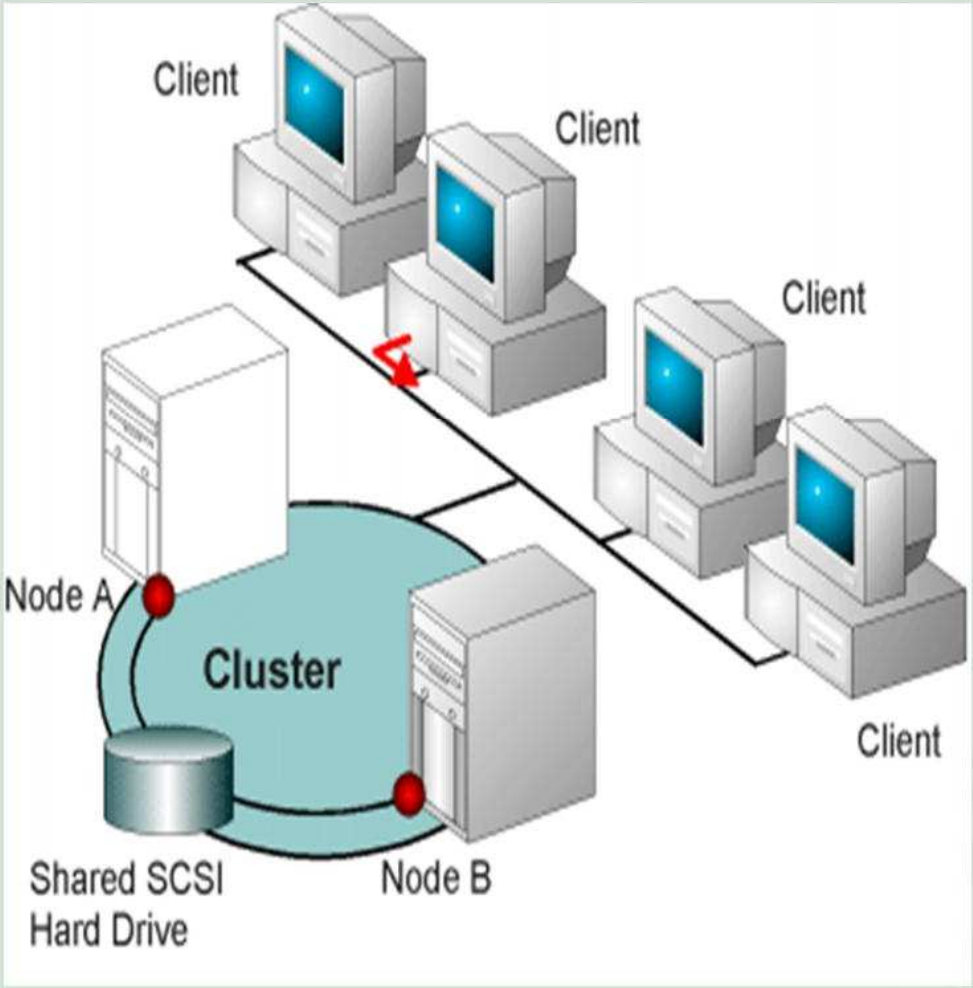
- Grouping multiple standalone Computers in a cluster by a network.
- The components of a cluster are connected to each other through ***fast local area networks***
- Many types of computer can be refer to cluster computers, starting from a poor-man's supercomputer to ***COWs*** (Clusters Of Workstations), and ***NOWs*** (Networks Of Workstations)

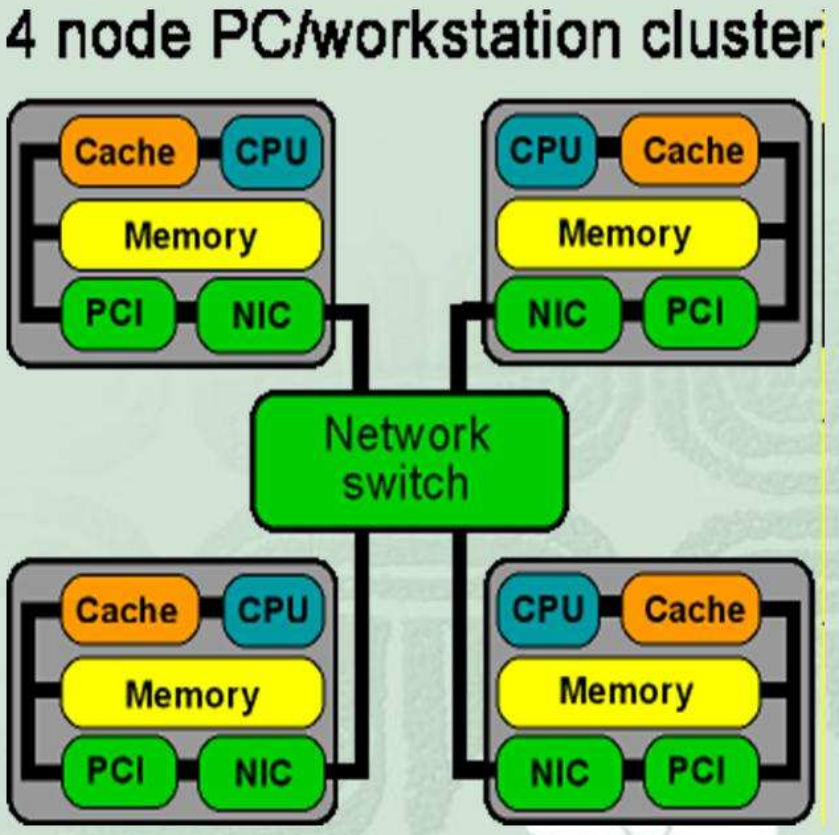
Cluster Architecture

For Parallel Processing (HPC)



Cluster Workstations





High Performance Computing (HPC) Cluster

- HPC clusters used where;
  - ❖ Time to solution a problem is important.
  - ❖ A problem is too big and can't fit on one single computer.
- Ideal to run many similar jobs with different parameters or data sets (**SPMD**)
  - Hundreds of jobs submit and allow the cluster to manage work flow.
  - ❖ Depending on resources, all jobs run simultaneously, or some may wait in queue while other jobs finish.
  - ❖ This type of computing is local to a cluster node, the node doesn't communicate with other nodes, but may need high speed file system access (**Farm Computing Model**).
- Ideal to run ***job*** consists of dependent ***tasks***
  - ✓ Concerning about tasks ***proceeding***

Back

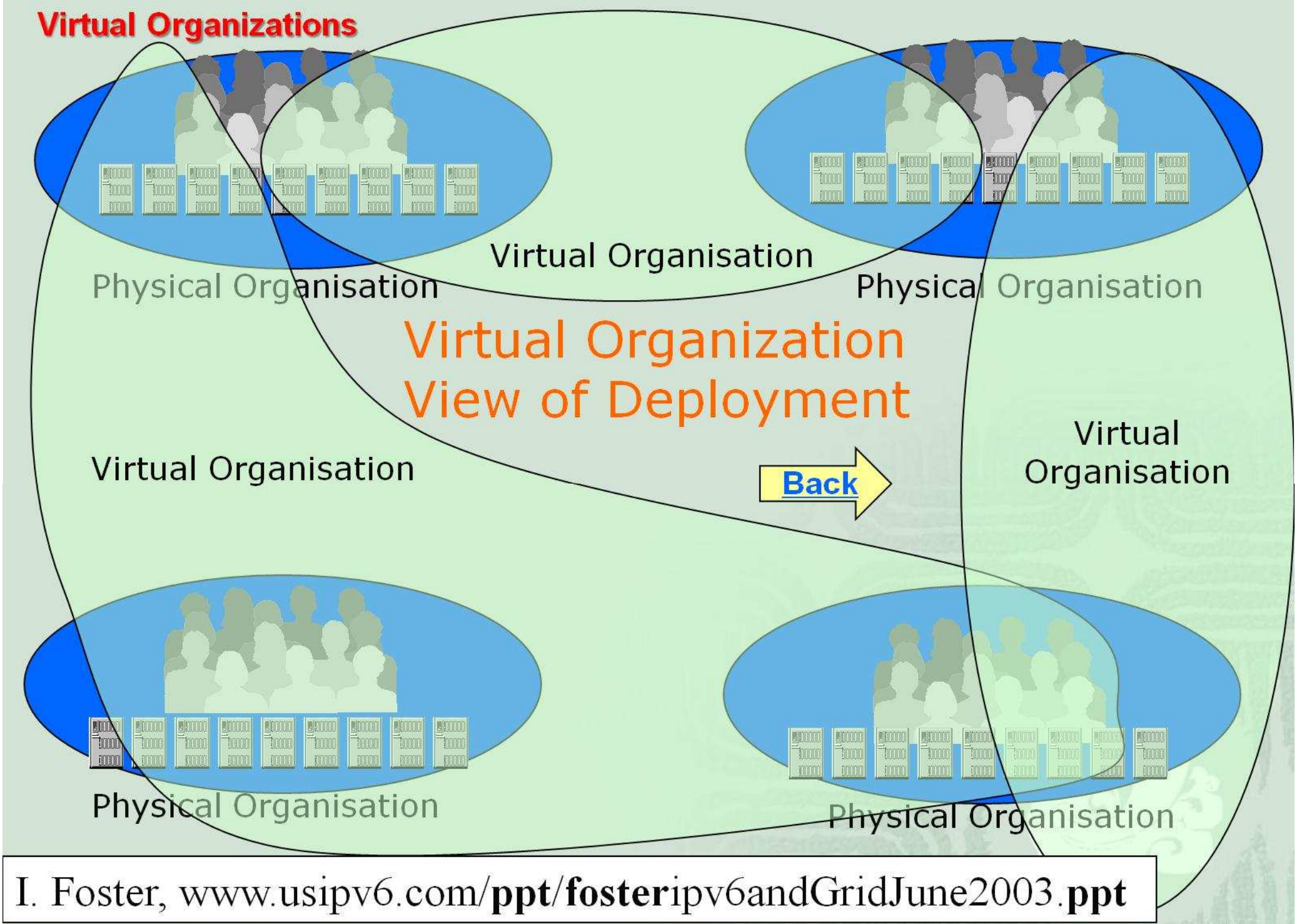
Utility Computing

- The design of Utility Computing based on a service providing
  - (i.e., ***business model***)
  - ➢ when the users (consumers) need computing resources, they ***pay*** providers for using it.
- Utility Computing classified as :
  - ➢ Grid Computing
  - ➢ Cloud Computing

Grid Computing

- Using computers communicating over the ***Internet*** to work on ***a given problem or given application***.
- Grid computing enables coordinated ***resource sharing and problem solving in dynamic, multi-institutional virtual organizations.***
  - Using P2P as infrastructure of Grid
- Enterprises or organizations present grids as integrated computing resources. They viewed as *virtual platforms* to support *virtual organizations*.
- The grids types
  - Knowledge, Data, Computational, Application Service Provisioning, Interaction or Utility.
- Most grid computing applications need middleware software to manage network resources

Why???



I. Foster, www.usipv6.com/**ppt**/**foster**ipv6andGridJune2003.**ppt**

Cloud Computing

- Will be discussed in details

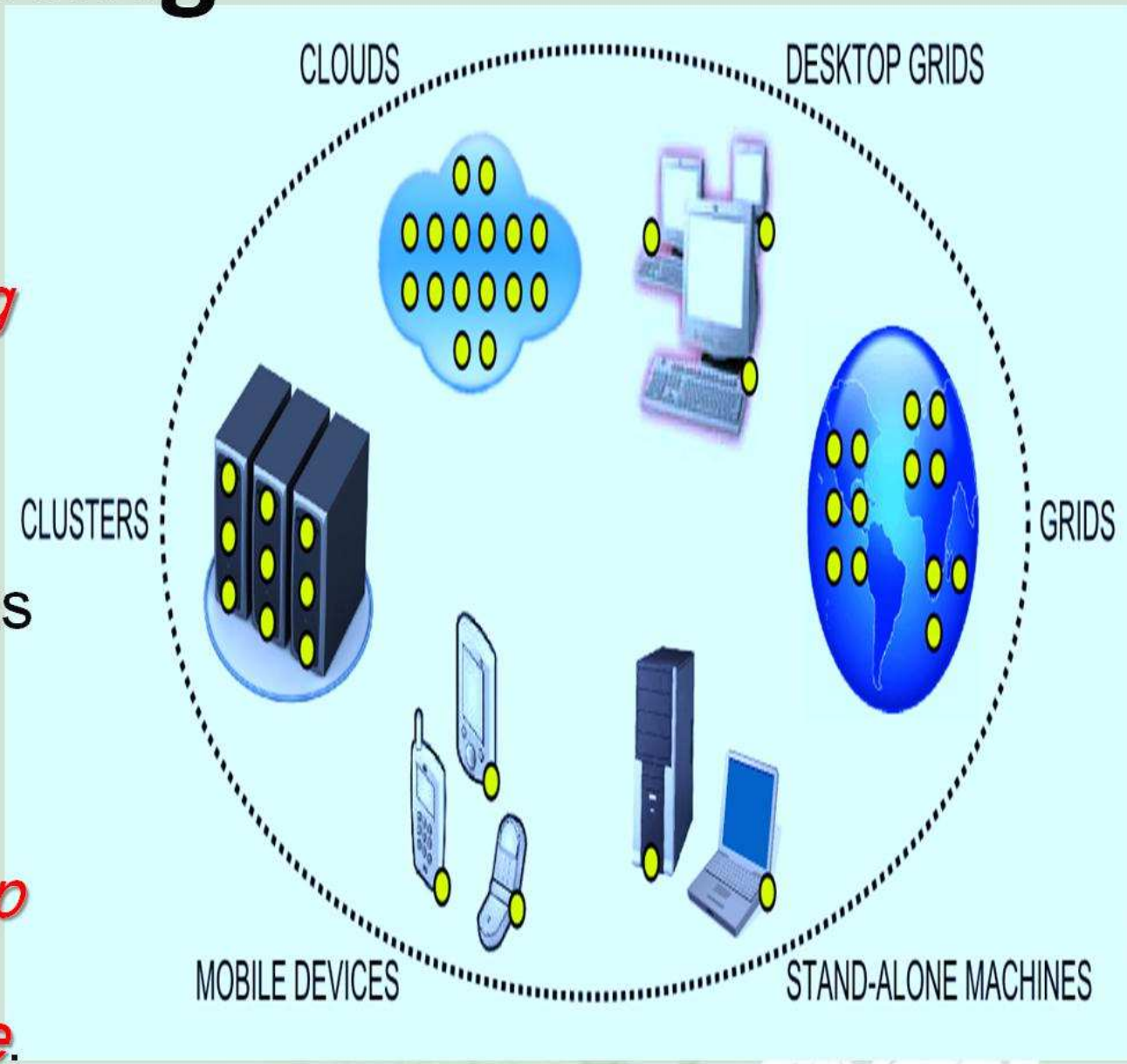
Comparison Between Grid and Cloud

	Grid Computing	Cloud Computing
Means of utilization (e.g. Harris 2008)	Allocation of multiple servers onto a single task or job	Virtualization of servers; one server to compute several tasks concurrently
Typical usage pattern (e.g. EGEE 2008)	Typically used for job execution, i.e. the execution of a program for a limited time	More frequently used to support long-running services
Level of abstraction (e.g. Jha et al. 2008)	Expose high level of detail	Provide higher-level abstractions

Back

Jungle Computing

- Combination of *heterogeneous*, *hierarchical*, and *distributed computing resources*.
- In many realistic scientific research areas, domain experts are being forced into concurrent use of multiple *clusters*, *grids*, *clouds*, *desktop grids*, *independent computers*, and *more*.



Jungle Computing

- Jungle computing refers to the ***use of diverse, distributed and highly non-uniform performance computer systems to achieve peak performance.***
- **Now**
  - Scientists can use grid and cloud infrastructures, in a variety of combinations along with traditional supercomputers - all connected via fast networks.
  - ***Peak performance computing*** use multiple diverse platforms and systems simultaneously, giving rise to the term "***jungle computing***".
- **Example**
  - –Ibis high-performance distributed programming system is an example of the jungle computing.