Parallel Distributed System

Dr. Bibhudatta Sahoo
Communication & Computing Group
Department of CSE, NIT Rourkela
Email: bdsahu@nitrkl.ac.in, 9937324437, 2462358

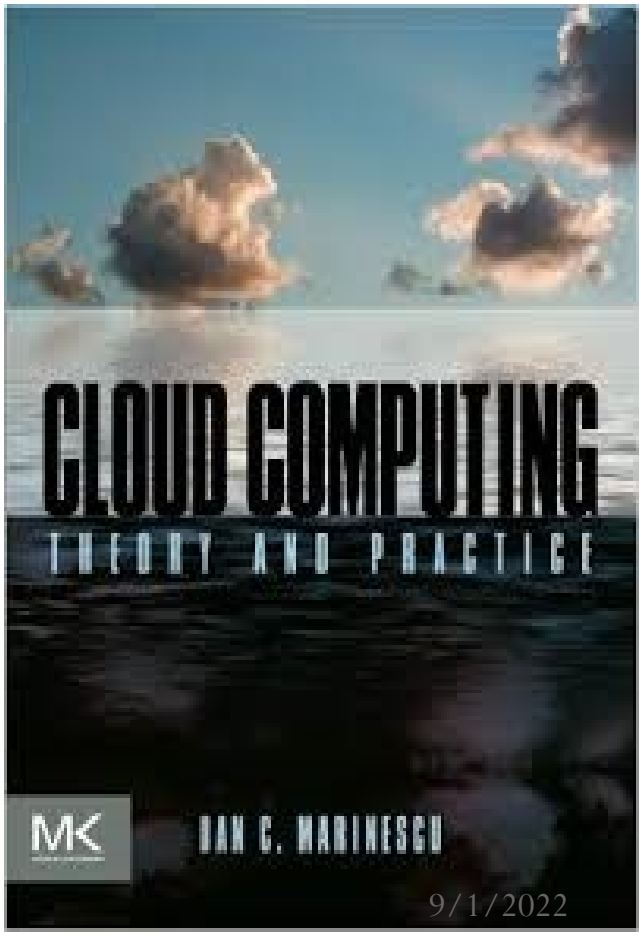
Cloud Computing: Theory and Practice

Dan C. Marinescu
Computer Science Division
Department of Electrical Engineering & Computer Science
University of Central Florida, Orlando, FL 32816, USA
Email: dcm@cs.ucf.edu

November 10, 2012

Chapter 2

Dan C. Marinescu *was a professor of computer science at Purdue University from 1984 to 2001. Then he joined the Computer Science Department at the University of Central Florida. He has held visiting faculty positions at the IBM T. J. Watson Research Center, the Institute of Information Sciences in Beijing, the Scalable Systems Division of Intel Corp., Deutsche Telecom AG and INRIA Rocquancourt in France. His research interests cover parallel and distributed systems, cloud computing, scientific computing, quantum computing, and quantum information theory.*



Chapter 2: Parallel and Distributed Systems

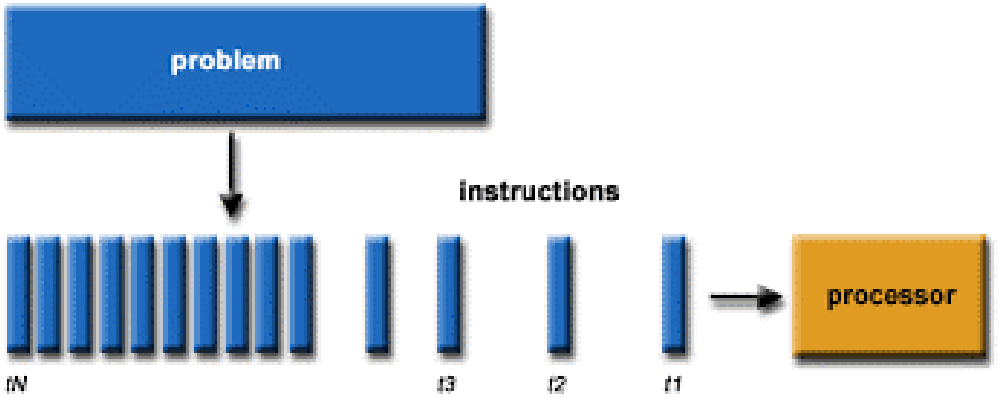
Parallel Computing

Parallel computing

- Parallel computing refers to the **process of breaking down larger problems into smaller, independent**, often similar parts that can be executed simultaneously by multiple processors communicating via shared memory, the results of which are combined upon completion as part of an overall algorithm.
- Parallel computing is a type of computing architecture in which several processors simultaneously execute multiple, smaller calculations broken down from an overall larger, complex problem.
- Parallel computing infrastructure is typically housed within a single datacenter where several processors are installed in a server rack; computation requests are distributed in small chunks by the application server that are then executed simultaneously on each server.

Serial computer and computing

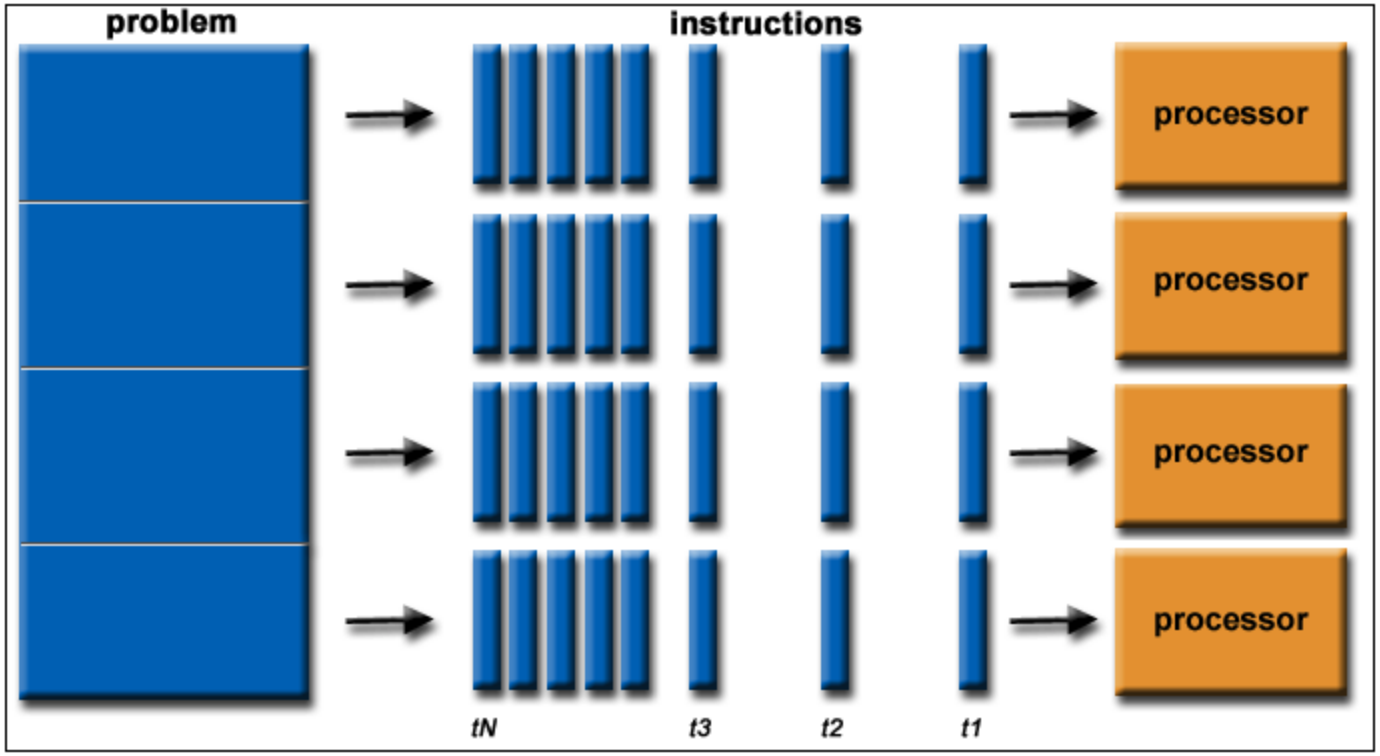
- A serial computer is **a computer typified by bit-serial architecture** – i.e., internally operating on one bit or digit for each clock cycle.



- Serial processing is **a type of processing in which one task is completed at a time and all the tasks are executed by the processor in a sequence.**

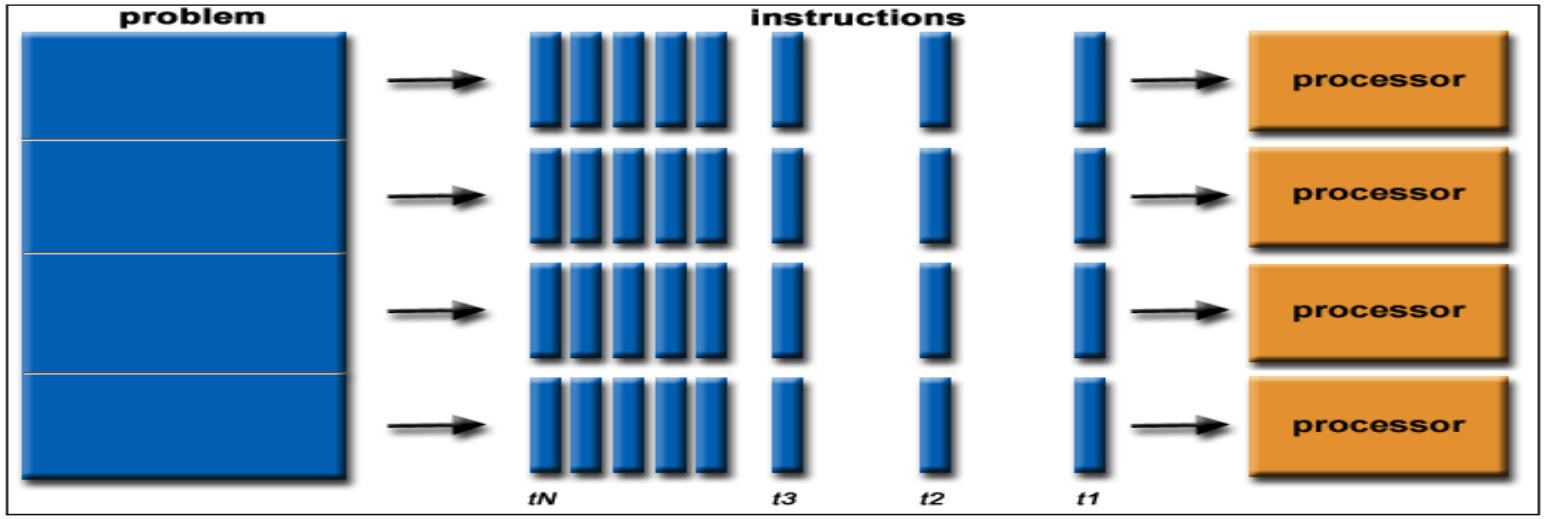
Parallel computing

- Parallel computing refers to the process of executing several processors an application or computation simultaneously.



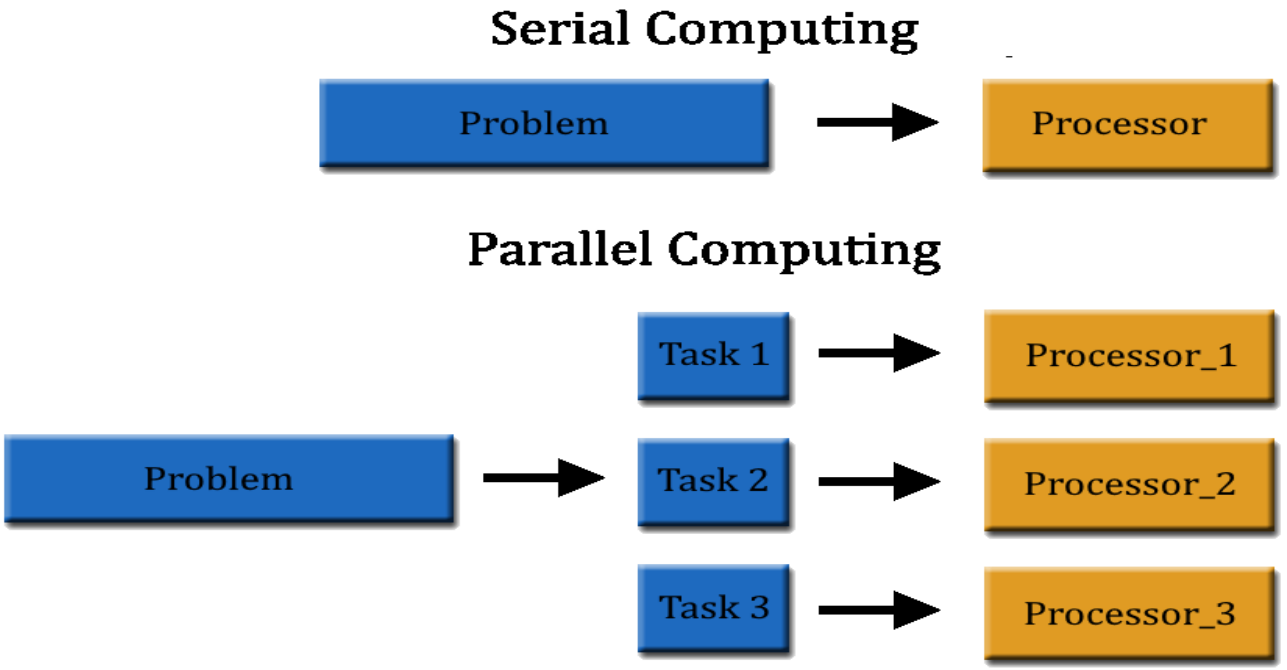
Parallel computing

- Generally, it is a kind of computing architecture where the large problems break into independent, smaller, usually similar parts that can be processed in one go.
- It is done by multiple CPUs communicating via shared memory, which combines results upon completion.
- It helps in performing large computations as it divides the large problem between more than one processor.



Parallel processing vs Serial processing

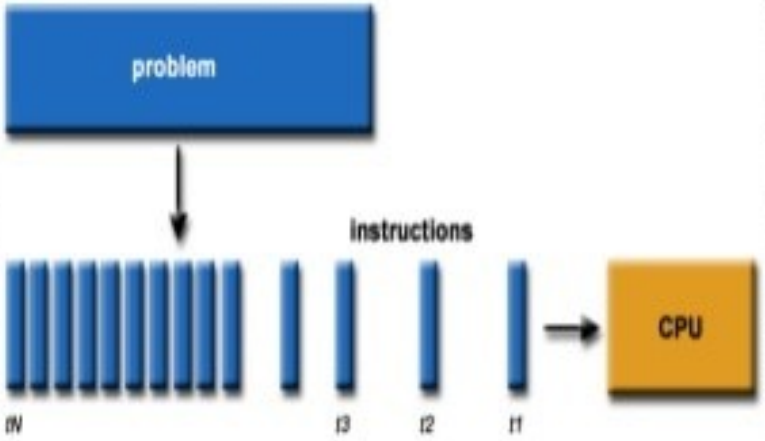
- Serial processing is a type of processing in which one task is completed at a time and all the tasks are executed by the processor in a sequence.
- Parallel processing is a type of processing in which multiple tasks are completed at a time by different processors.



Parallel Vs Serial Computing

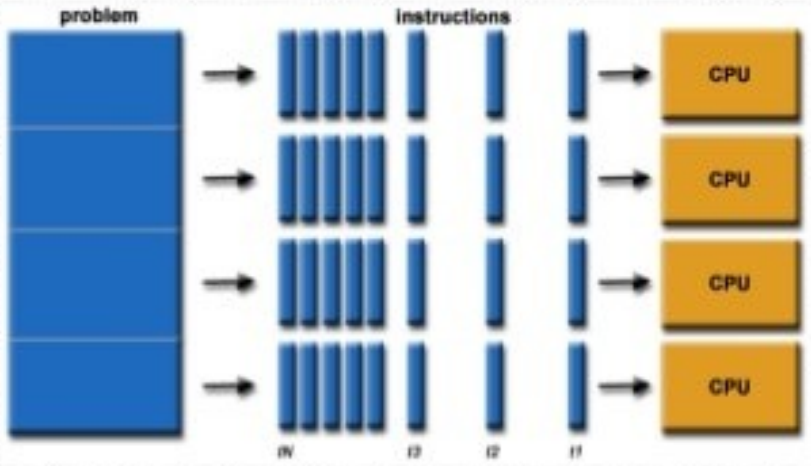
Serial Computation

- To Run on a single computer with single CPU
- Problem is broken into discrete instructions and are executed one after another
- One instruction at a time



Parallel Computation

- To Run on Multiple CPUs
- Problem is broken into discrete parts that can be solved concurrently
- Each part is broken down into instruct-ions which is executed simultaneously on different CPUs



Advantages of Parallel computing

- In parallel computing, more resources are used to complete the task that led to decrease the time and cut possible costs. Also, cheap components are used to construct parallel clusters.
- Comparing with Serial Computing, parallel computing can solve larger problems in a short time.
- For simulating, modeling, and understanding complex, real-world phenomena, parallel computing is much appropriate while comparing with serial computing.
- When the local resources are finite, it can offer benefit you over non-local resources.
- There are multiple problems that are very large and may impractical or impossible to solve them on a single computer; the concept of parallel computing helps to remove these kinds of issues.
- One of the best advantages of parallel computing is that it allows you to do several things in a time by using multiple computing resources.
- Furthermore, parallel computing is suited for hardware as serial computing wastes the potential computing power.

Disadvantages of Parallel Computing

- It addresses Parallel architecture that can be difficult to achieve.
- In the case of clusters, better cooling technologies are needed in parallel computing.
- It requires the managed algorithms, which could be handled in the parallel mechanism.
- The multi-core architectures consume high power consumption.
- The parallel computing system needs low coupling and high cohesion, which is difficult to create.
- The code for a parallelism-based program can be done by the most technically skilled and expert programmers.
- Although parallel computing helps you out to resolve computationally and the data-exhaustive issue with the help of using multiple processors, sometimes it affects the conjunction of the system and some of our control algorithms and does not provide good outcomes due to the parallel option.
- Due to synchronization, thread creation, data transfers, and more, the extra cost sometimes can be quite large; even it may be exceeding the gains because of parallelization.
- Moreover, for improving performance, the parallel computing system needs different code tweaking for different target architectures

Fundamentals of Parallel Computer Architecture

- Parallel Computer Architecture is **the method of organizing all the resources to maximize the performance and the programmability within the limits given by technology** and the cost at any instance of time.
- Parallel computer architecture is classified on the basis of the level at which the hardware supports parallelism. There are different classes of parallel computer architectures, which are as follows:

1. **Multi-core computing**
2. **Symmetric multiprocessing**
3. **Distributed Computing**
4. **Massively parallel computing**

Interconnect

Different ways in which we can connect CPUs (Computing Units) together.

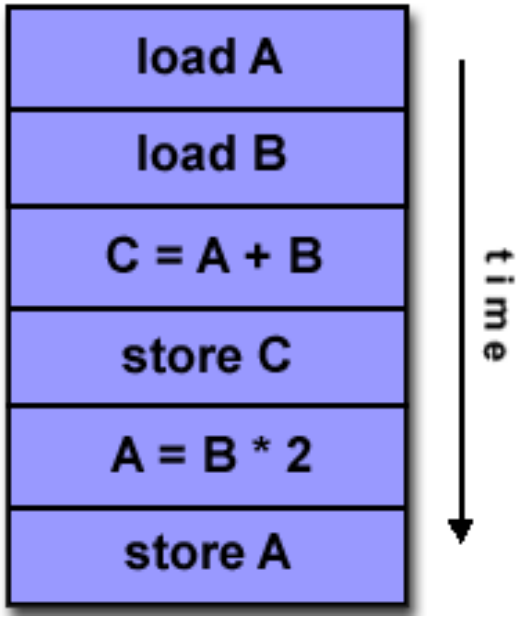
Interconnect (Flynn's Taxonomy)

- There are different ways in which we can connect CPUs together. The most widely used classification scheme (taxonomy) is that created by Flynn in 1972. It classifies machines by the number of **instruction streams** and the number of **data streams**.
- An **instruction stream** refers to the sequence of instructions that the computer processes.
- **Multiple instruction streams** means that different instructions can be executed concurrently.
- **Data streams** refer to memory operations

Flynn's Taxonomy

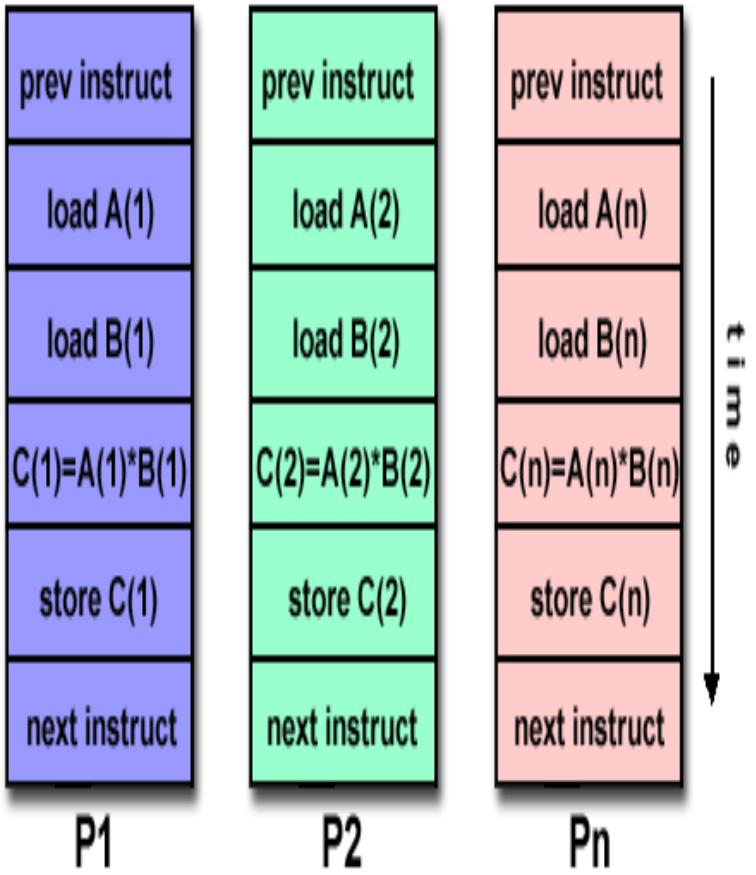
- **SISD:** Single instruction stream, single data stream. This is the traditional uniprocessor computer.
- **SIMD:** Single instruction stream, multiple data streams. This is an array processor; a single instruction operates on many data units in parallel.
- **MISD:** Having multiple concurrent instructions operating on a single data element makes no sense. This isn't a useful category.
- **MIMD:** Multiple instruction stream, multiple data streams. This is a broad category covering all forms of machines that contain multiple computers, each with a program counter, program, and data. It covers **parallel** and **distributed systems.**

Single Instruction, Single Data (SISD):



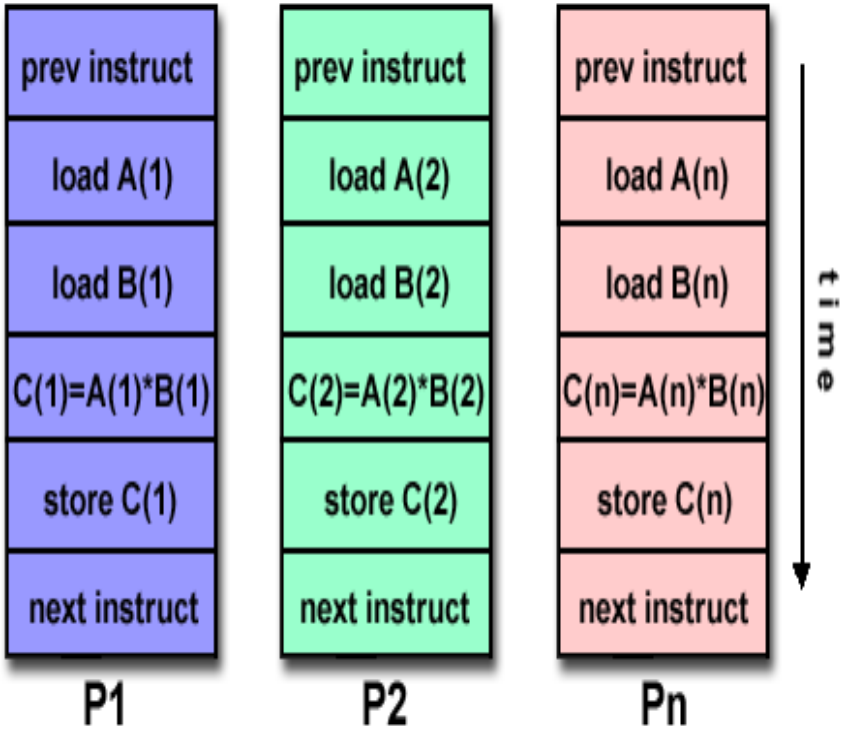
- A serial (non-parallel) computer
- **Single Instruction:** Only one instruction stream is being acted on by the CPU during any one clock cycle
- **Single Data:** Only one data stream is being used as input during any one clock cycle
- Deterministic execution
- This is the oldest and even today, the most common type of computer

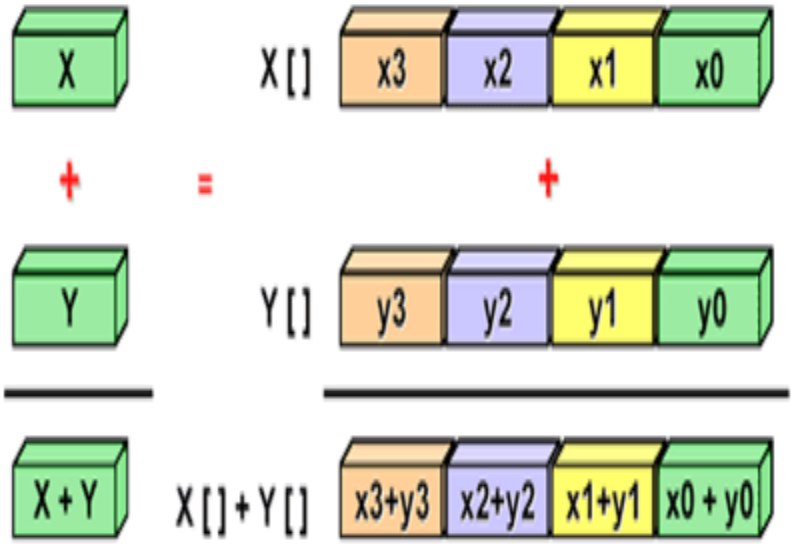
Single Instruction, Multiple Data (SIMD):



- A type of parallel computer
- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Best suited for specialized problems characterized by a high degree of regularity, such as graphics/image processing.
- Synchronous (lockstep) and deterministic execution
- Two varieties: Processor Arrays and Vector Pipelines

SIMD

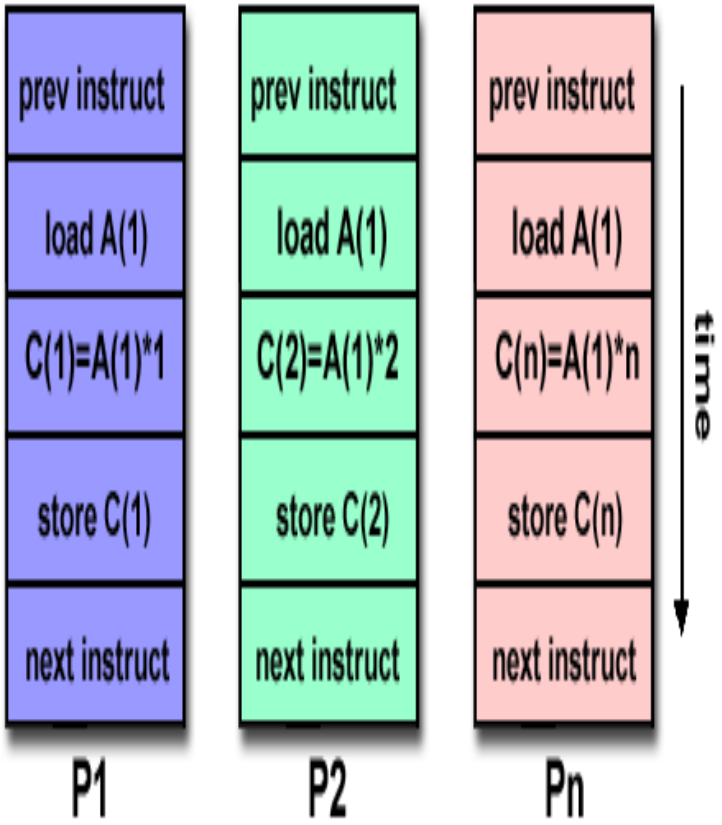




Examples:

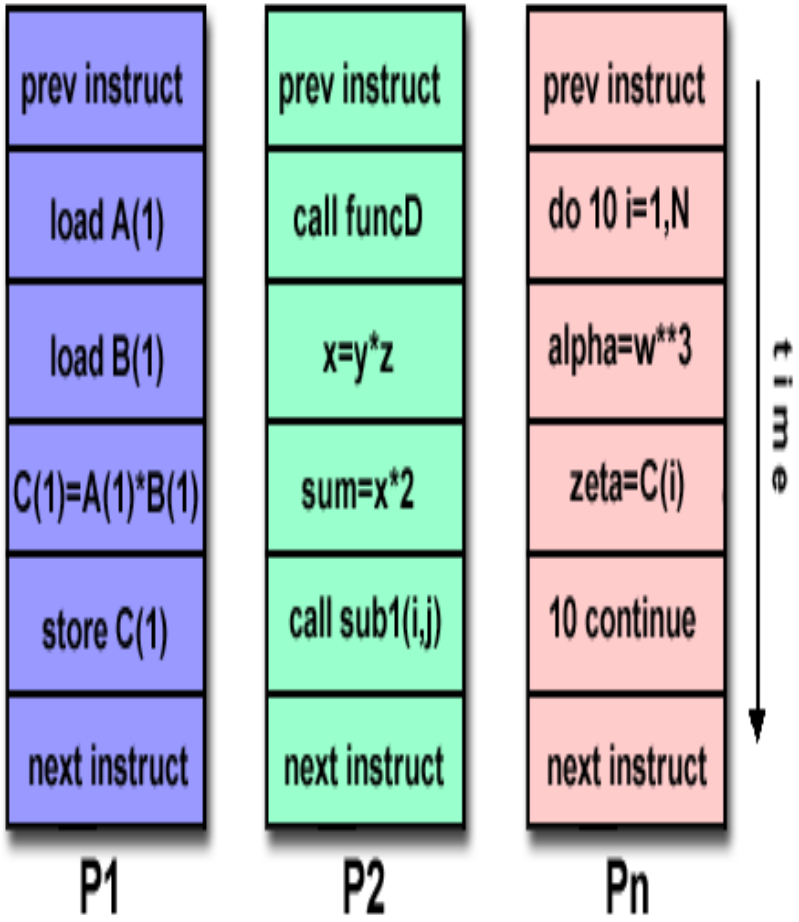
Processor Arrays: Connection Machine CM-2, MasPar MP-1 & MP-2, ILLIAC IV
Vector Pipelines: IBM 9000, Cray X-MP, Y-MP & C90, Fujitsu VP, NEC SX-2, Hitachi S820, ETA10

Multiple Instruction, Single Data (MISD):



- A type of parallel computer
- **Multiple Instruction:** Each processing unit operates on the data independently via separate instruction streams.
- **Single Data:** A single data stream is fed into multiple processing units.
- Few actual examples of this class of parallel computer have ever existed. One is the experimental Carnegie-Mellon C.mmp computer (1971).
- Some conceivable uses might be:
  - multiple frequency filters operating on a single signal stream
  - multiple cryptography algorithms attempting to crack a single coded message.

Multiple Instruction, Multiple Data (MIMD):



- A type of parallel computer
- **Multiple Instruction:** Every processor may be executing a different instruction stream
- **Multiple Data:** Every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Currently, the most common type of parallel computer - most modern supercomputers fall into this category.
- **Examples:** most current supercomputers, networked parallel computer clusters and "grids", multi-processor SMP computers, multi-core PCs.
- Note: many MIMD architectures also include SIMD execution sub-components

Classifications of MIMD

- Memory
- Interconnect network
- Coupling

Memory

- We refer to machines with shared memory as multiprocessors and to machines without shared memory as multicomputers.
- A multiprocessor contains a single virtual address space.
- If one processor writes to a memory location, we expect another processor to read the value from that same location.
- A multicomputer is a system in which each machine has its own memory and address space.

Interconnection network

- Machines can be connected by either a bus or a switched network. On a bus, a single network, bus, or cable connects all machines.
- The bandwidth on the interconnection is shared.
- On a switched network, individual connections exist between machines, guaranteeing the full available bandwidth between machines.

Coupling

- A **tightly-coupled system** is one where the components tend to be reliably connected in close proximity. It is characterized by short message delays, high bandwidth, and high total system reliability.
- A **loosely-coupled system** is one where the components tend to be distributed. Message delays tend to be longer and bandwidth tends to be lower than in closely-coupled systems.
- Reliability expectations are that individual components may fail without affecting the functionality of other components

Distributed Computing System

Defining a distributed system

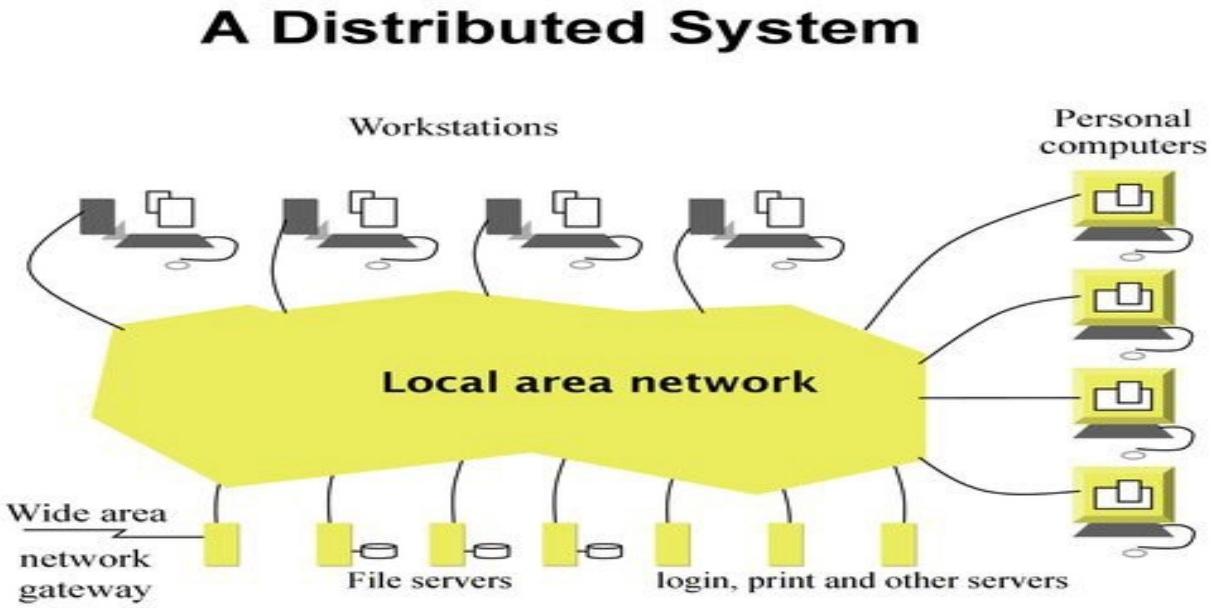
- **Andrew Tannenbaum** defines a distributed system as a "collection of ***independent*** computers that appear to the users of the ***system as a single computer***."

- **independent**: This means that, architecturally, the machines are capable of operating independently. T
- **single system image** : The software enables this set of connected machines to appear as a single computer to the users of the system.

- This major goal in designing distributed systems that are easy to maintain and operate

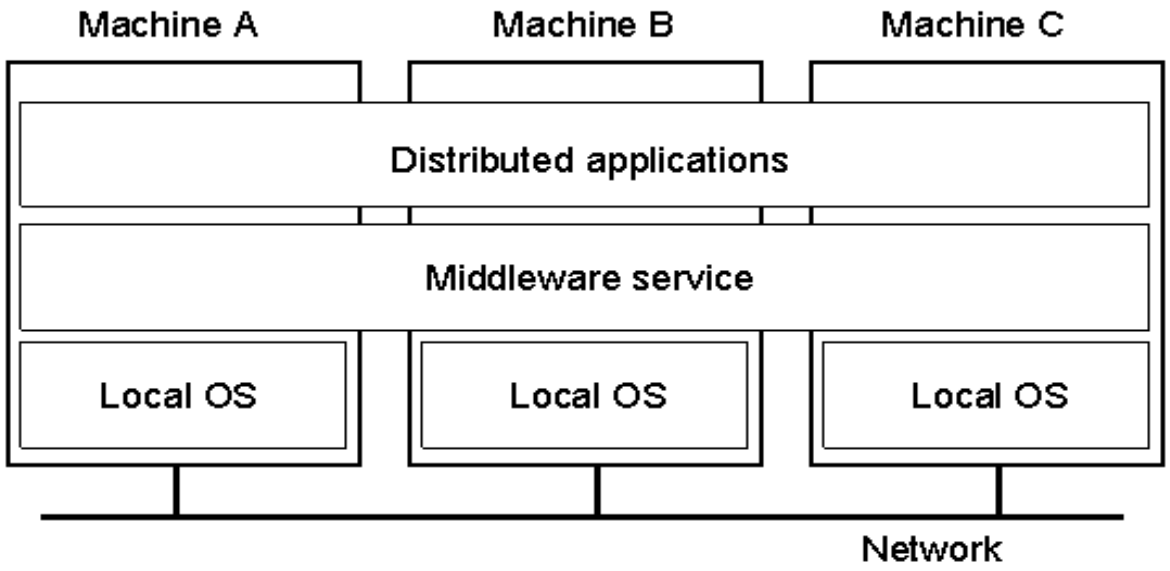
Defining Distributed Systems

- A distributed system is one in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing.
- Definition above covers the entire range of distributed systems in which networked computers can usefully be deployed.



Definition of a Distributed System

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



- A distributed system organized as middleware. Note that the middleware layer extends over multiple machines

Distributed systems

- Distributed systems involve the interaction between disparate independent entities, bounded by common language and protocols and working toward a common goal.

Examples

- Different types of distributed systems are found in real life. One of the biggest and perhaps the most complex distributed system is human society itself.
- In the digital world, the Internet has become a very important distributed environment for everybody.

General Examples of Distributed Systems

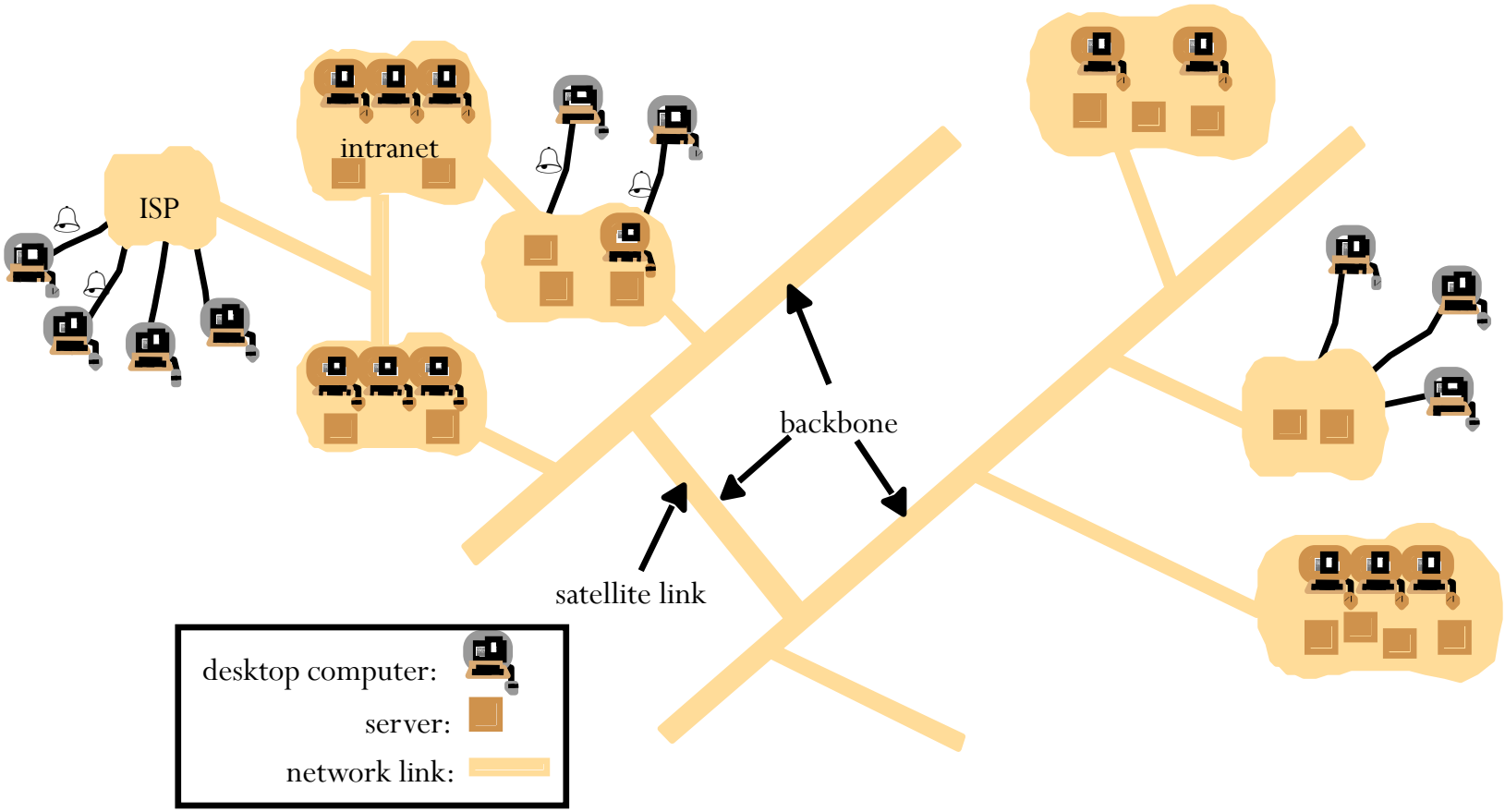


Figure 1. A typical portion of the Internet

General Examples of Distributed Systems

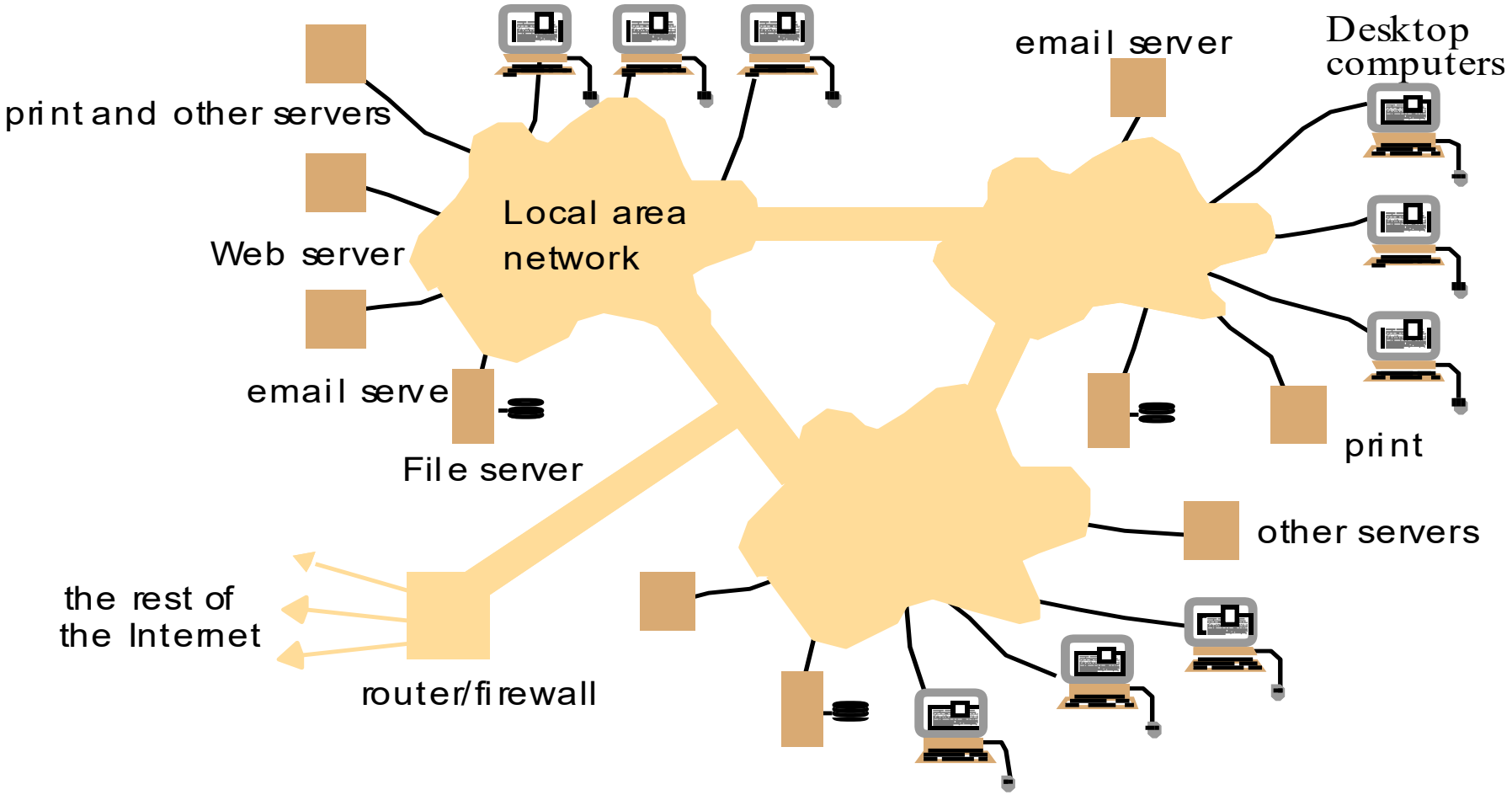


Figure 2. A typical Interanet

Resource sharing and the Web

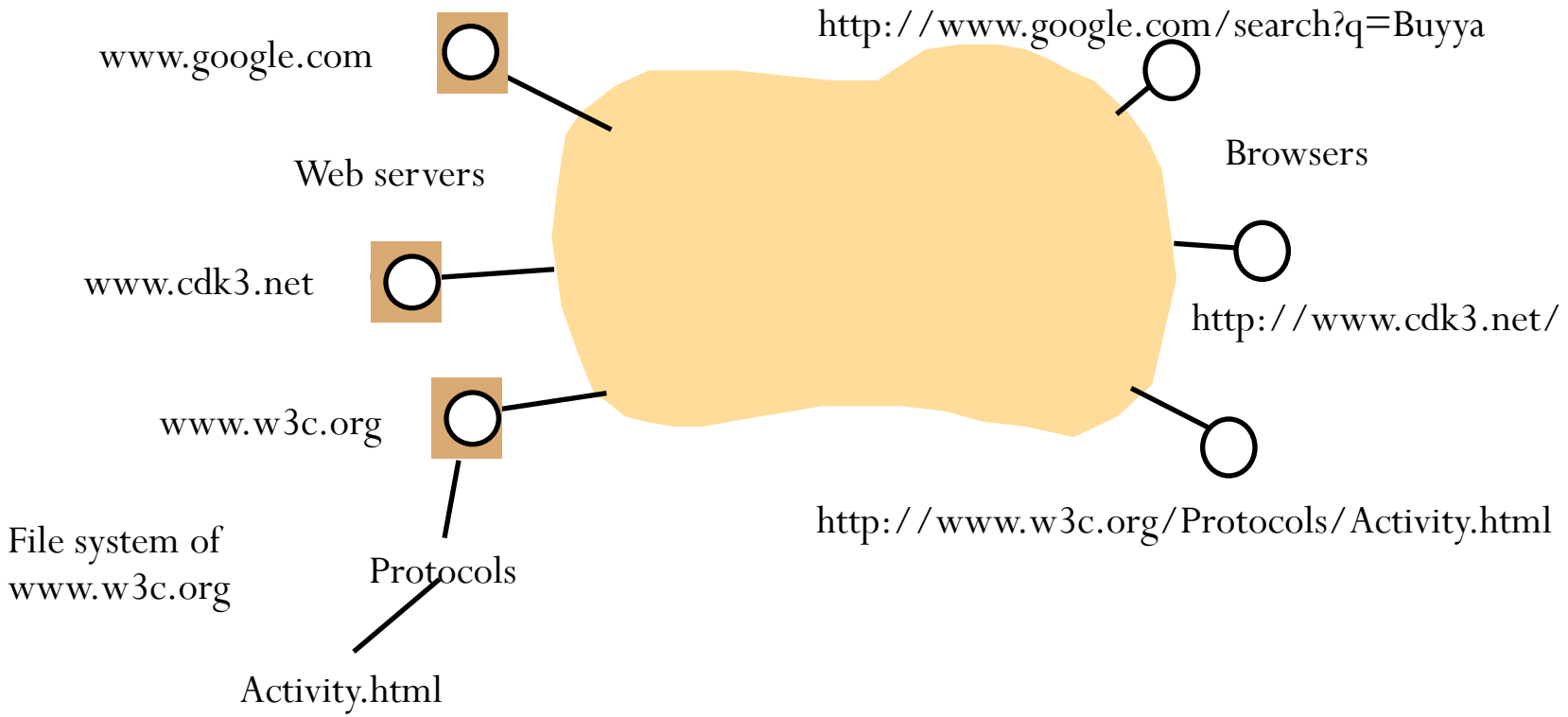


Figure 4 : **Web servers and web browsers.**

Advantages of Distributed Systems over Centralized Systems

- **Economics**: a collection of microprocessors offer a better price/performance than mainframes. Low price/performance ratio: cost effective way to increase computing power.

- **Speed**: a distributed system may have more total computing power than a mainframe. Ex. 10,000 CPU chips, each running at 50 MIPS. Not possible to build 500,000 MIPS single processor since it would require 0.002 nsec instruction cycle. Enhanced performance through load distributing.

- **Inherent distribution**: Some applications are inherently distributed. Ex. a supermarket chain.

Advantages of Distributed Systems over Centralized Systems

- **Reliability**: If one machine crashes, the system as a whole can still survive. Higher availability and improved reliability.
- **Incremental growth**: Computing power can be added in small increments. Modular expandability
- **Another deriving force**: the existence of large number of personal computers, the need for people to collaborate and share information.

Characteristics of Distributed Systems

(1) Multiple entities: *One of the key characteristics of a distributed system is the* presence of multiple – in many cases a great many – entities participating in the system.

- The entities can be users or subsystems which compose the distributed system.

(2) Heterogeneity: *Another key characteristic is the heterogeneous nature of the* entities involved.

- The heterogeneity may lie in the type of system or user, underlying policies and/or the data/resources that the underlying subsystems consume.
- The heterogeneity of distributed systems can be best observed in the Internet, where multitudes of systems, protocols, policies and environments interact to create a scalable infrastructure.

Characteristics of Distributed Systems

(3) Concurrency: Another important characteristic that distinguishes any distributed system from a centralized one is concurrency.

- Different components of distributed systems may run concurrently as the components may be loosely coupled. Therefore there is a need to understand the synchronization issues during the design of distributed systems.

(4) Resource sharing: *Sharing of resources is another key characteristic of distributed* systems.

Desirable Characteristics of Distributed Systems

- (1) *Openness: A desirable characteristic for a distributed system is openness of* the underlying architecture, protocols, resources and infrastructure, where they can be extended or replaced without affecting the system behavior. If we look at the Internet, this issue is nicely handled through the use of open standards: we can see the interplay between different protocols, standards, infrastructures and architectures without affecting the activities of the Internet as a whole.
- (2) *Scalability: One of the key motivations for going from a centralized system* to a distributed one is to increase the overall scalability of the system. Hence to have a highly scalable system is desirable in any form of distributed system.

Desirable Characteristics of Distributed Systems

- (3) *Transparency: Another desirable characteristic is to have transparency in the* operation. From the user's and the subsystem's point of view, the underlying systems should be transparent. Primarily, transparency can be of two types – *location transparency and system transparency. The first type talks* about the need to be transparent regarding the location disparity between different systems. The second talks about the need to be transparent about system issues like failure, concurrency, scaling, migration and so on.

Types of Distributed Systems

- Distributed systems can be divided into mainly three types: distributed computing systems, distributed information systems and distributed pervasive systems.

1. **Distributed computing systems** is mainly concerned with providing computations in a distributed manner.
2. **Distributed information systems** is mainly concerned with providing information in a distributed manner.
3. **Distributed pervasive systems** is the next-generation distributed system, which is ubiquitous in nature

Design Challenges of Distributed Systems

Heterogeneity: standards and protocols; middleware; virtual machine;

Openness: publication of services; notification of interfaces;

***Security*:** firewalls; encryption;

***Scalability*:** replication; caching; multiple servers;

Failure Handling: failure tolerance; recover/roll-back; redundancy;

***Concurrency*:** concurrency control to ensure data consistency;

***Transparency*:** middleware; location transparent naming; anonymity

Design Challenges of Distributed Systems

1. **Heterogeneity:** Heterogeneous components must be able to interoperate.

2. **Openness:** Interfaces should allow components to be added or replaced.

3. **Security:** The system should only be used in the way intended.

Design Challenges of Distributed Systems

4. **Scalability:** System should work efficiently with an increasing number of users. System performance should increase with inclusion of additional resources.

5. **Failure handling:** Failure of a component (partial failure) should not result in failure of the whole system.

6. **Concurrency:** concurrency control to ensure data consistency;

7. **Transparency:** Distribution should be hidden from the user as much as possible.

1:Heterogeneity

- The internet enables users to access services over a variety and difference:
  - Networks.
  - Operating systems.
  - Implementations by different developers.
- The different networks are masked by the fact that all of the attached computers are communicate to each other using standard internet protocols.
- The *middleware* software layers (like CORBA) provide a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, operating systems and programming languages.
- *Virtual machines* approach provides a way of making code executable on any hardware – *mobile code*.

2: Openness

- The openness of distributed systems is determined by the degree to which new resource-sharing services can be added and be available for use by variety of client programs (*services publication*).
- The specification and documentation of key software interfaces of the system components must be available to software developers (*interfaces notification*).
- Systems that are designed to support resource sharing in this way are termed open distributed systems to emphasize the fact that they are extensible.

3: Security

- The security of many information resources available and maintained in distributed systems is importance.
- Security for information resources has three components:
  - Confidentiality: protection against disclosure to unauthorized individuals.
  - Integrity: protection against alteration or corruption.
  - Availability: protection against interference with the means to access the resources.
- A *firewall* can be used to form a barrier around an intranet to protect it from outside users but does not deal with ensuring the appropriate use of resources by users within the intranet.
- *Encryption* can be used to provide adequate protection of shared resources and to keep sensitive information secret when is transmitted in messages over the internet.
- Receiving of an executable program (mobile code) as an electronic mail attachment needs to be handled with care – effects of running the program are unpredictable!

4: Scalability

- 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.

Date	*Computers*	*Web servers*
1979, Dec.	188	0
1989, July	130,000	0
1999, July	56,218,000	5,560,866

- The design of scalable dist. systems presents many challenges:
  - Controlling the cost of physical resources.
  - Controlling the performance loss.
  - Preventing software resources running out.
  - Avoiding performance bottlenecks.

5: Failure Handling

- Failures in a distributed system are partial, therefore the handling of it is particularly difficult.
- Some failures can be detected but others are difficult or impossible to detect it.
- Some detected failures can be hidden or made less severe.

5: Failure Handling

- Recovery from failures involves the design of software so that the state of permanent data can be rolled back after a server has crashed.
- Services can be made to tolerate failures by the use of redundant components:
  - At least two different routes between any two routers in the internet.
  - Databases and domain name tables must be replicated in several servers.

6: Concurrency

- Several clients in a distributed system can access a shared resource at the same time
- Any object that represents a shared resource in a distributed system (a programmer that implement it) must be responsible for ensuring that it operate correctly in a concurrent environment.
- Operations of objects in a concurrent environment must be synchronized in a way that its data remains consistent.
- The synchronization of an object operations can be achieved by standard techniques such as semaphores.

7: Transparency

- *Access transparency*: enables local and remote resources to be accessed using identical operations.

- *Location transparency*: enables resources to be accessed without knowledge of their location.

- *Concurrency transparency*: enables several processes to operate concurrently using shared resources without interference between them.

7: Transparency

- *Replication transparency*: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.

- *Failure transparency*: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.

- *Mobility transparency*: allows the movement of resources and clients within a system without affecting the operation of users or programs.

7: 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.
- Access and location transparency together provide *network transparency*.

Applications of distributed computing and newer challenges

Distributed Systems

- Early distributed systems emphasized the single system image – often tried to make a networked set of computers look like an ordinary general purpose computer
- Examples: Amoeba, Sprite, NOW, Condor (distributed batch system), Distributed systems run distributed applications, from file sharing to large scale projects like SETI@Home http://setiathome.ssl.berkeley.edu/

1: Mobile systems

- Mobile systems typically use wireless communication which is based on electromagnetic waves and utilizes a shared broadcast medium.
- The characteristics of communication are different; many issues such as range of transmission and power of transmission come into play, besides various engineering issues such as battery power conservation, interfacing with the wired Internet, signal processing and interference.
- From a computer science perspective, there is a rich set of problems such as **routing, location management**, **channel allocation**, **localization** and **position estimation**, and the overall **management of mobility**.

There are two popular architectures for a mobile network

- The first is the ***base-station*** approach, also known as the *cellular approach*, wherein a *cell* which is the geographical region within range of a static but powerful base transmission station is associated with that base station.
- All mobile processes in that cell communicate with the rest of the system via the base station.
- The second approach is the ***ad-hoc network*** approach where there is no base station (which essentially acted as a centralized node for its cell).
- All responsibility for communication is distributed among the mobile nodes, wherein mobile nodes have to participate in routing by forwarding packets of other pairs of communicating nodes.

2: Sensor networks

- A sensor is a processor with an electro-mechanical interface that is capable of sensing physical parameters, such as temperature, velocity, pressure, humidity, and chemicals.
- Recent developments in cost-effective hardware technology have made it possible to deploy very large (of the order of 106 or higher) low-cost sensors.
- An important paradigm for monitoring distributed events is that of event streaming, which was defined earlier.
- The streaming data reported from a sensor network differs from the streaming data reported by "computer processes" in that the events reported by a sensor network are in the environment, external to the computer network and processes.

2: Sensor networks

- This limits the nature of information about the reported event in a sensor network. Sensor networks have a wide range of applications.
- Sensors may be mobile or static; sensors may communicate wirelessly, although they may also communicate across a wire when they are statically installed.
- Sensors may have to self-configure to form an ad-hoc network, which introduces a whole new set of challenges, such as position estimation and time estimation.

3: Ubiquitous or pervasive computing

- Ubiquitous systems represent a class of computing where the processors embedded in and seamlessly pervading through the environment perform application functions in the background, much like in sci-fi movies.
- The intelligent home, and the smart workplace are some example of ubiquitous environments currently under intense research and development.
- Ubiquitous systems are essentially distributed systems; recent advances in technology allow them to leverage wireless communication and sensor and actuator mechanisms.
- They can be self-organizing and network-centric, while also being resource constrained. Such systems are typically characterized as having many small processors operating collectively in a dynamic ambient network.

4:Peer-to-peer computing

- Peer-to-peer (P2P) computing represents computing over an application layer network wherein all interactions among the processors are at a "peer" level, without any hierarchy among the processors.
- All processors are equal and play a symmetric role in the computation. P2P computing arose as a paradigm shift from client–server computing where the roles among the processors are essentially asymmetrical.
-

4: Peer-to-peer computing

- P2P networks are typically self-organizing, and may or may not have a regular structure to the network.
- No central directories (such as those used in domain name servers) for name resolution and object lookup are allowed. Some of the key challenges include: object storage mechanisms, efficient object lookup, and retrieval in a scalable manner; dynamic reconfiguration with nodes as well as objects joining and leaving the network randomly; replication strategies to expedite object search; tradeoffs between object size latency and table sizes; anonymity, privacy, and security.

Publish-subscribe, content distribution, and multimedia

- With the explosion in the amount of information, there is a greater need to receive and access only information of interest.
- Such information can be specified using filters.
- In a dynamic environment where the information constantly fluctuates (Example: varying stock prices),
- there needs to be:
- (i) an efficient mechanism for distributing this information (***publish***),
- (ii) an efficient mechanism to allow end users to indicate interest in receiving specific kinds of information (***subscribe***), *and*
- *(iii)* an efficient mechanism for ***aggregating*** large volumes of published information and filtering it as per the user's subscription filter

Publish-subscribe, content distribution, and multimedia

- Content distribution refers to a class of mechanisms, primarily in the web and P2P computing context, whereby specific information which can be broadly characterized by a set of parameters is to be distributed to interested processes.
- Clearly, there is overlap between content distribution mechanisms and publish–subscribe mechanisms. When the content involves multimedia data, special requirement such as the following arise: multimedia data is usually very large and information-intensive, requires compression, and often requires special synchronization during storage and playback.

Distributed agents

- Agents are software processes or robots that can move around the system to do specific tasks for which they are specially programmed.
- The name "agent" derives from the fact that the agents do work on behalf of some broader objective.
- Agents collect and process information, and can exchange such information with other agents.
- Often, the agents cooperate as in an ant colony, but they can also have friendly competition, as in a free market economy.
- Challenges in distributed agent systems include coordination mechanisms among the agents, controlling the mobility of the agents, and their software design and interfaces.

Distributed data mining

- Data mining algorithms examine large amounts of data to detect patterns and trends in the data, to *mine or extract useful information*.
- *A traditional* example is: examining the purchasing patterns of customers in order to profile the customers and enhance the efficacy of directed marketing schemes.
- The mining can be done by applying database and artificial intelligence techniques to a data repository.
- In many situations, the data is necessarily distributed and cannot be collected in a single repository, as in banking applications where the data is private and sensitive, or in atmospheric weather prediction where the data sets are far too massive to collect and process at a single repository in real-time. In such cases, efficient distributed data mining algorithms are required

Security in distributed systems

- The traditional challenges of security in a distributed setting include: **confidentiality** (ensuring that only authorized processes can access certain information), **authentication** (ensuring the source of received information and the identity of the sending process), and **availability** (maintaining allowed access to services despite malicious actions).
- The goal is to meet these challenges with **efficient** and **scalable** solutions.
- These basic challenges have been addressed in traditional distributed settings for the newer distributed architectures, such as wireless, peer-to-peer, grid, and pervasive computing.
- Security issues due to a **resource-constrained environment**, a **broadcast medium**, the **lack of structure**, and the **lack of trust in the network.**

Grid Computing Systems

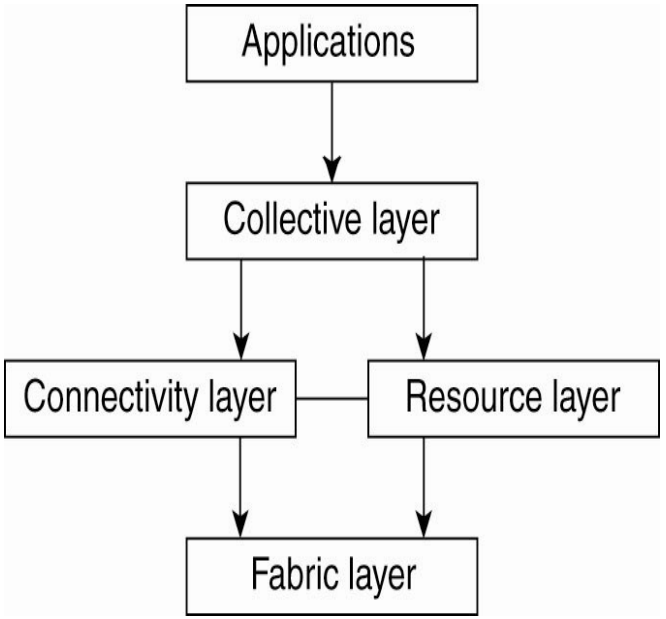
- Modeled loosely on the electrical grid.
- Highly heterogeneous with respect to hardware, software, networks, security policies, etc.
- Grids support **virtual organizations**: a collaboration of users who pool resources (servers, storage, databases) and share them
- Grid software is concerned with managing sharing across administrative domains.

Grids

- Similar to clusters but processors are more loosely coupled, tend to be heterogeneous, and are not all in a central location.
- Can handle workloads similar to those on supercomputers, but grid computers connect over a network (Internet?) and supercomputers' CPUs connect to a high-speed internal bus/network
- Problems are broken up into parts and distributed across multiple computers in the grid – less communication between parts than in clusters.

A Proposed Architecture for Grid Systems*

- **Fabric layer**: interfaces to local resources at a specific site
- **Connectivity layer**: protocols to support usage of *multiple resources* for a single application; e.g., access a remote resource or transfer data between resources; and protocols to provide security
- **Resource layer** manages a *single resource,* using functions supplied by the connectivity layer
- **Collective layer:** resource discovery, allocation, scheduling, etc.
- **Applications**: use the grid resources
- The collective, connectivity and resource layers together form the middleware layer for a grid



A layered architecture for grid computing systems

Cloud computing

- Provides scalable services as a utility over the Internet.
- Often built on a computer grid
- Users buy services from the cloud
  - Grid users may develop and run their own software
- Cluster/grid/cloud distinctions blur at the edges!

Transparency

- One of the main goals of a distributed operating system is to make the existence of multiple computers invisible (transparent) and provide a single system image to its users. That is, a distributed operating system must be designed in such a way that a collection of distinct machines connected by a communication subsystem appears to its users as a virtual uniprocessor.
- Achieving complete transparency is a difficult task and requires that several different aspects of transparency be supported by the distributed operating system.
- The eight forms of transparency identified by the International Standards Organization's Reference Model for Open Distributed Processing [ISO 1992] are **access transparency**, **location transparency, replication transparency, failure transparency, migration transparency, concurrency transparency, performance transparency,** and **scaling transparency.**

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 may be 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

Different forms of transparency in a distributed system

Transparency in a Distributed System

Access Transparency

- Access transparency means that users should not need or be able to recognize whether a resource (hardware or software) is remote or local. This implies that the distributed operating system should allow users to access remote resources in the same way as local resources.

Location Transparency

- The two main aspects of location transparency are as follows:
- 1. Name transparency. This refers to the fact that the name of a resource (hardware or software) should not reveal any hint as to the physical location of the resource.
- 2. User mobility. This refers to the fact that no matter which machine a user is logged onto, he or she should be able to access a resource with the same name.

Transparency in a Distributed System

Migration Transparency

- For better performance, reliability, and security reasons, an object that is capable of being moved (such as a process or a file) is often migrated from one node to another in a distributed system.

Transparency in a Distributed System

Replication Transparency

- For better performance and reliability, almost all distributed operating systems have the provision to create replicas (additional copies) of files and other resources on different nodes of the distributed system.
- In these systems, both the existence of multiple copies of a replicated resource and the replication activity should be transparent to the users.

Failure Transparency

- Failure transparency deals with masking from the users' partial failures in the system, such as a communication link failure, a machine failure, or a storage device crash.
- A distributed operating system having failure transparency property will' continue to function, perhaps in a degraded form, in the face of partial failures.

Transparency in a Distributed System

Migration Transparency

- For better performance, reliability, and security reasons, an object that is capable of being moved (such as a process or a file) is often migrated from one node to another in a distributed system.

Concurrency Transparency

- Concurrency transparency means that each user has a feeling that he or she is the sole user of the system and other users do not exist in the system.

Performance Transparency

- The aim of performance transparency is to allow the system to be automatically reconfigured to improve performance, as loads vary dynamically in the system.

Scaling Transparency

- The aim of scaling transparency is to allow the system to expand in scale without disrupting the activities of the users.

Suggested Reading

- **Jean Dollimore, Tim Kindberg, George Coulouris** , "Distributed Systems: Concepts and Design," Addison-Wesley 2005.
- **Ajay D. Kshemkalyani, Mukesh Singhal**, "Distributed Computing: Principles, Algorithm, and Systems", Cambridge university press 2008.
- **Andrew Tannenbaum, Maarten van Steen** , "Distributed Systems: Principles and Paradigms," Prentice Hall (2nd edition) 2006.
- **Nancy Lynch** , "Distributed Algorithms," Morgan Kaufmann 1996.
- **Hagit Attiya, Jennifer, Welch,** *Distributed Computing: Fundamentals, Simulations, and Advanced Topics.Wiley-Interscience. ISBN 0471453242.*
- **Mukesh Singhal, Shivaratri**, *"Advanced Concepts in Operating System", Tata McGraw-Hill, 2001*

Suggested Reading