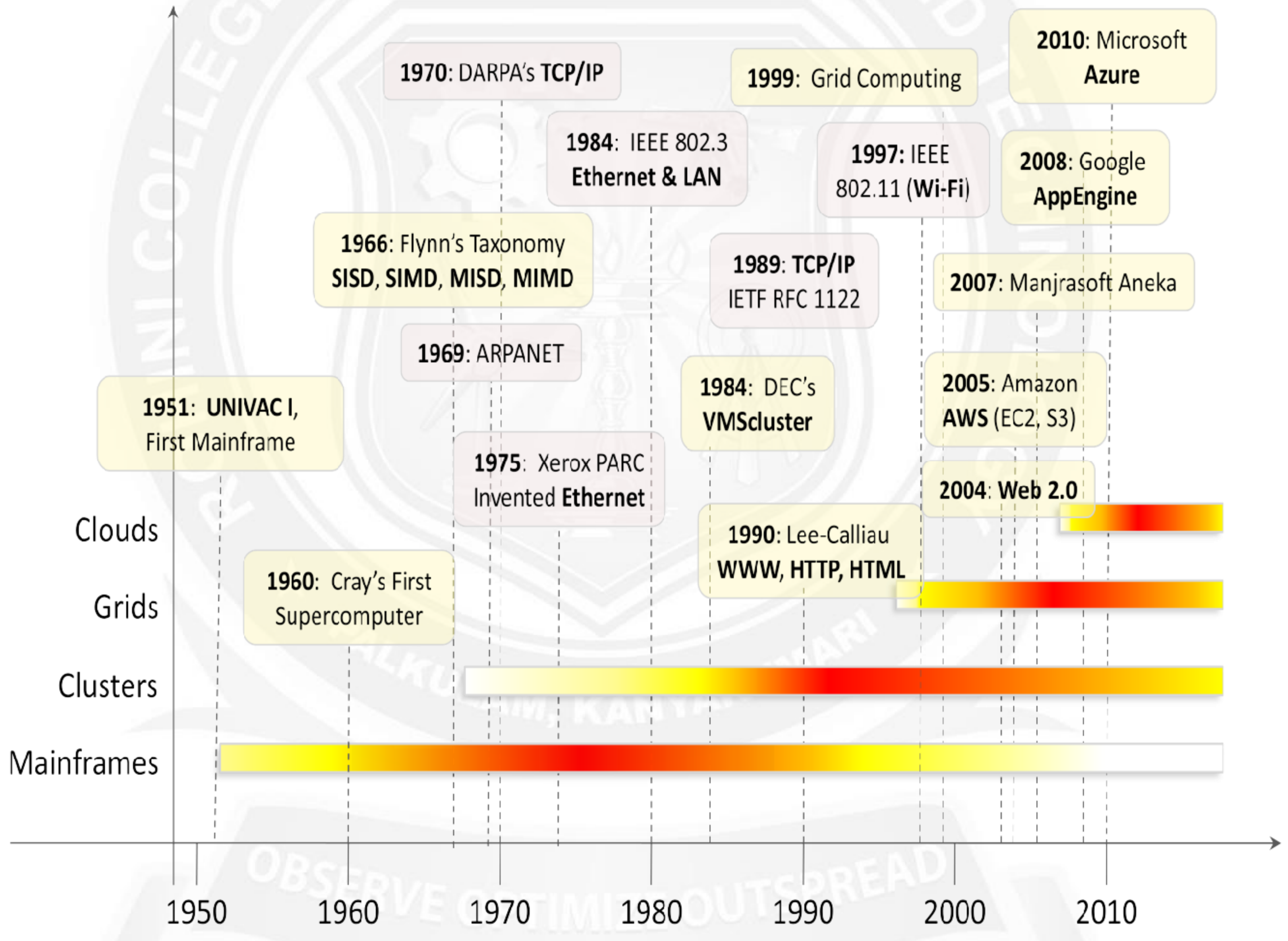
1.4 Principles of Parallel and Distributed Computing

- **Three major milestones have led to cloud computing evolution**
  - Mainframes: Large computational facilities leveraging multiple processing units. Even though mainframes cannot be considered as distributed systems, they offered large computational power by using multiple processors, which were presented as a single entity to users.

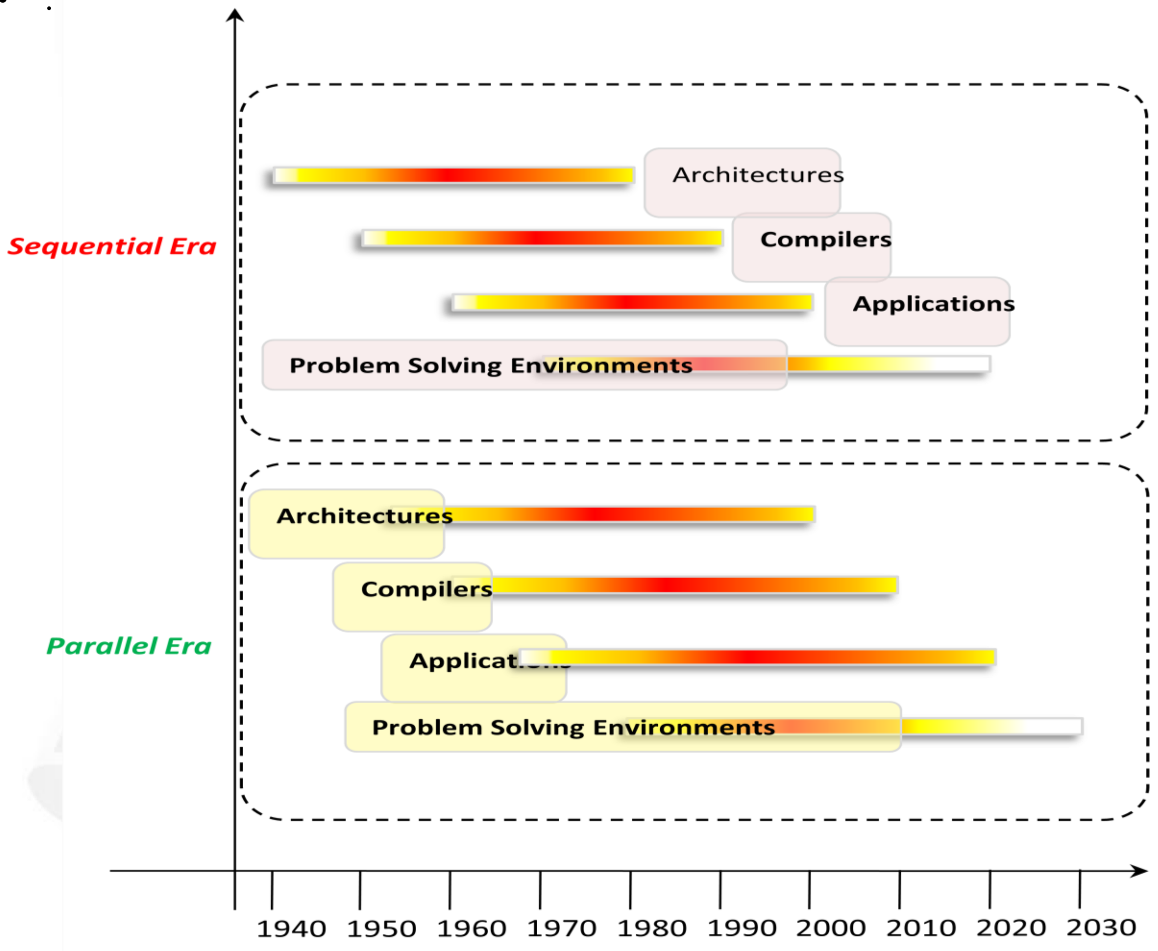
Mile Stones to Cloud computing Evolution


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
1984: DEC's VMScluster
2005: Amazon AWS (EC2, S3)
1951: UNIVAC I, First Mainframe
1975: Xerox PARC Invented Ethernet
2004: Web 2.0
1990: Lee-Calliau WWW, HTTP, HTML
1960: Cray's First Supercomputer
Clouds
Grids
Clusters
Mainframes
1950
1960
1970
1980
1990
2000
2010

- Clusters: An alternative technological advancement to the use of mainframes and super computers.
- Grids
- Clouds

1.4.1 Eras of Computing

- Two fundamental and dominant models of computing are ***sequential*** and ***parallel***.
  - The sequential era began in the 1940s, and Parallel(and distributed) computing era followed it within a decade.
- Four key elements of computing developed during three eras are
  - Architecture
  - Compilers
  - Applications
  - Problem solving environments
- .



- The computing era started with development in ***hardware architectures***, which actually enabled the creation of ***system software*** – particularly in the area of **compilers and operating systems** – which support the management of such systems and the development of ***applications***

- The term parallel computing and distributed computing are **often used interchangeably**, even though they mean **slightly different things**.
- The term **parallel implies a tightly coupled system**, where as **distributed systems refers to a wider class of system, including those that are tightly coupled**.
- More precisely, the term **parallel computing** refers to a model in which **the computation is divided among several processors sharing the same memory**.
- The architecture of **parallel computing system** is often characterized by the **homogeneity of components**: **each processor is of the same type and it has the same capability as the others.**

- The shared memory has a single address space, which is accessible to all the processors.
- Parallel programs are then broken down into several units of execution that can be allocated to different processors and can communicate with each other by means of shared memory.
- Originally parallel systems are considered as those architectures that featured multiple processors sharing the same physical memory and that were considered a single computer.
  - Over time, these restrictions have been relaxed, and parallel systems now include all architectures that are based on the concept of shared memory, whether this is physically present or created with the support of libraries, specific hardware, and a highly efficient networking infrastructure.
  - For example: a cluster of which of the nodes are connected through an InfiniBand network and configured with distributed shared memory system can be considered as a parallel system.
- The term distributed computing encompasses any architecture or system that allows the computation to be broken down into units and executed concurrently on different

computing elements, whether these are processors on different nodes, processors on the same computer, or cores within the same processor.

- Distributed computing includes a wider range of systems and applications than parallel computing and is often considered a more general term.
- Even though it is not a rule, the term distributed often implies that the locations of the computing elements are not the same and such elements might be heterogeneous in terms of hardware and software features.
- Classic examples of distributed computing systems are
  - Computing Grids
  - Internet Computing Systems

1.4.2 Elements of Parallel computing

- Silicon-based processor chips are reaching their physical limits. Processing speed is constrained by the speed of light, and the density of transistors packaged in a processor is constrained by thermodynamics limitations.
- A viable solution to overcome this limitation is to connect multiple processors working in coordination with each other to solve "Grand Challenge" problems.
- The first step in this direction led
  - To the development of parallel computing, which encompasses techniques, architectures, and systems for performing multiple activities in parallel.

a.Parallel Processing

- Processing of multiple tasks simultaneously on multiple processors is called ***parallel processing***.
- The parallel program consists of multiple active processes (tasks) simultaneously solving a given problem.
- A given task is divided into multiple subtasks using a divide-and-conquer technique, and each subtask is processed on a different central processing unit (CPU).
- Programming on multi processor system using the divide-and-conquer technique is called ***parallel programming***.
- Many applications today require more computing power than a traditional sequential computer can offer.

- Parallel Processing provides a cost effective solution to this problem by increasing the number of CPUs in a computer and by adding an efficient communication system between them.
- The workload can then be shared between different processors. This setup results in higher computing power and performance than a single processor a system offers.

Parallel Processing influencing factors

- The development of parallel processing is being influenced by many factors. The prominent among them include the following:
  - Computational requirements are ever increasing in the areas of both scientific and business computing. The technical computing problems, which require high-speed computational power, are related to
    - life sciences, aerospace, geographical information systems, mechanical design and analysis etc.
  - Sequential architectures are reaching mechanical physical limitations as they are constrained by the speed of light and thermodynamics laws.
    - The speed which sequential CPUs can operated is reaching saturation point (no more vertical growth), and hence an alternative way to get high computation speed is to connect multiple CPUs (opportunity for horizontal growth).
  - Hardware improvements in pipelining , super scalar, and the like are non scalable and require sophisticated compiler technology.
    - Developing such compiler technology is a difficult task.
  - Vector processing works well for certain kinds of problems. It is suitable mostly for scientific problems (involving lots of matrix operations) and graphical processing.
    - It is not useful for other areas, such as databases.
  - The technology of parallel processing is mature and can be exploited commercially
    - here is already significant R&D work on development tools and environments.
  - Significant development in networking technology is paving the way for

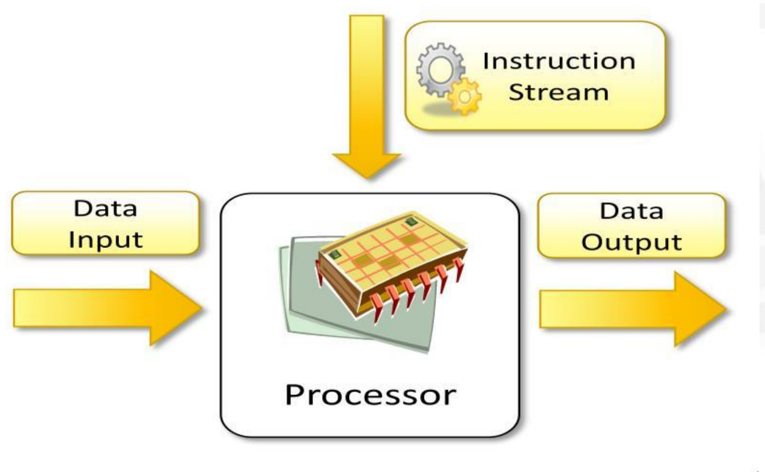
- heterogeneous computing.

b.Hardware architectures for parallel Processing

- The core elements of parallel processing are CPUs. Based on the number of instructions and data streams, that can be processed simultaneously, computing systems are classified into the following four categories:
  - Single-instruction, Single-data (SISD) systems
  - Single-instruction, Multiple-data (SIMD) systems
  - Multiple-instruction, Single-data (MISD) systems
  - Multiple-instruction, Multiple-data (MIMD) systems

(i) Single – Instruction , Single Data (SISD) systems

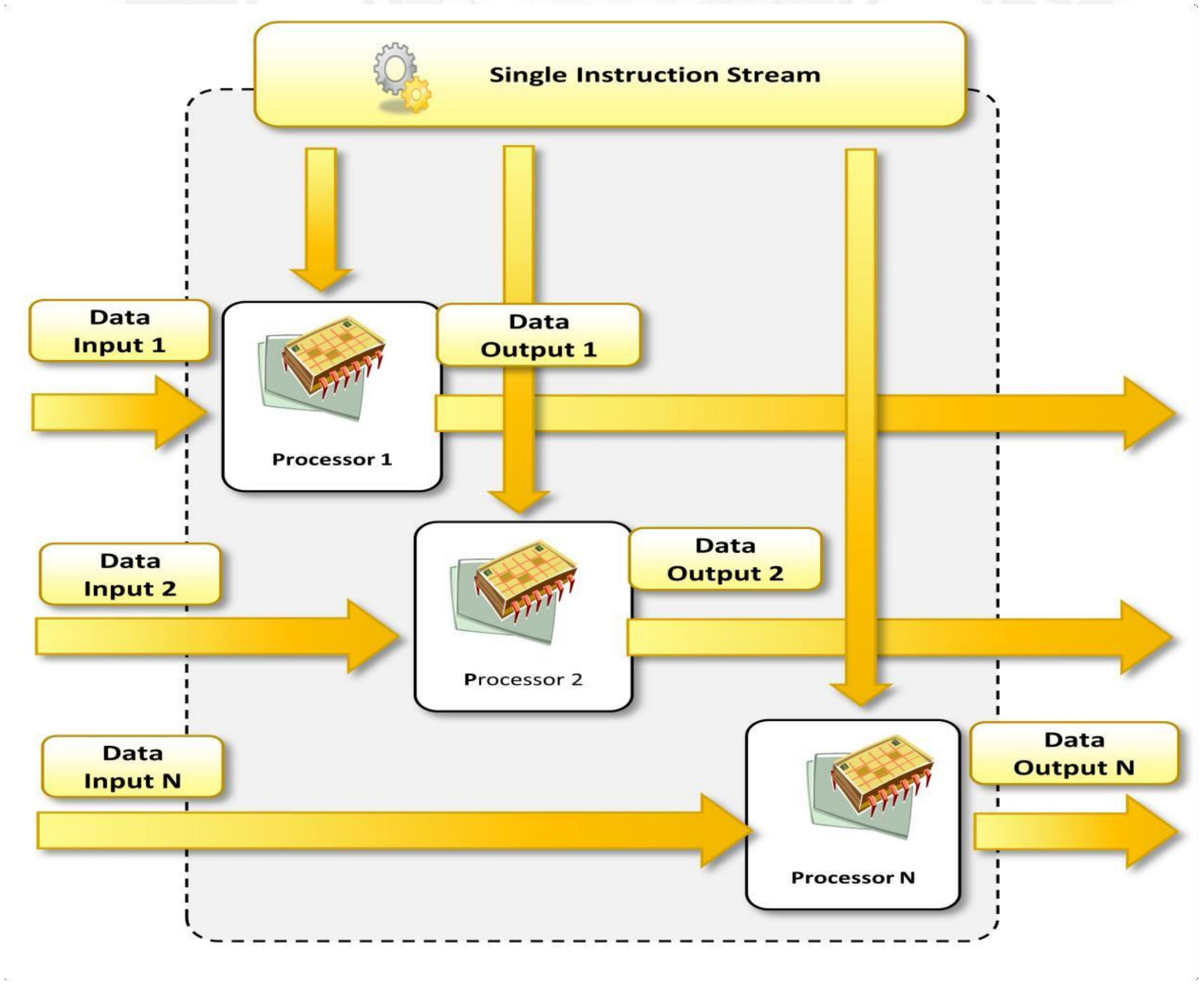
- SISD computing system is a uni-processor machine capable of executing a single instruction, which operates on a single data stream.
- Machine instructions are processed sequentially, hence computers adopting this model are popularly called sequential computers.
- Most conventional computers are built using SISD model.
- All the instructions and data to be processed have to be stored in primary memory.
- The speed of processing element in the SISD model is limited by the rate at which the computer can transfer information internally.
- Dominant representative SISD systems are IBM PC, Macintosh, and workstations.



(ii) Single – Instruction , Multiple Data (SIMD) systems

- SIMD computing system is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams.
- Machines based on this model are well suited for scientific computing since they involve lots of vector and matrix operations.
- For instance statement Ci = Ai * Bi, can be passed to all the processing elements (PEs), organized data elements of vectors A and B can be divided into multiple sets (N- sets for N PE systems), and each PE can process one data set.

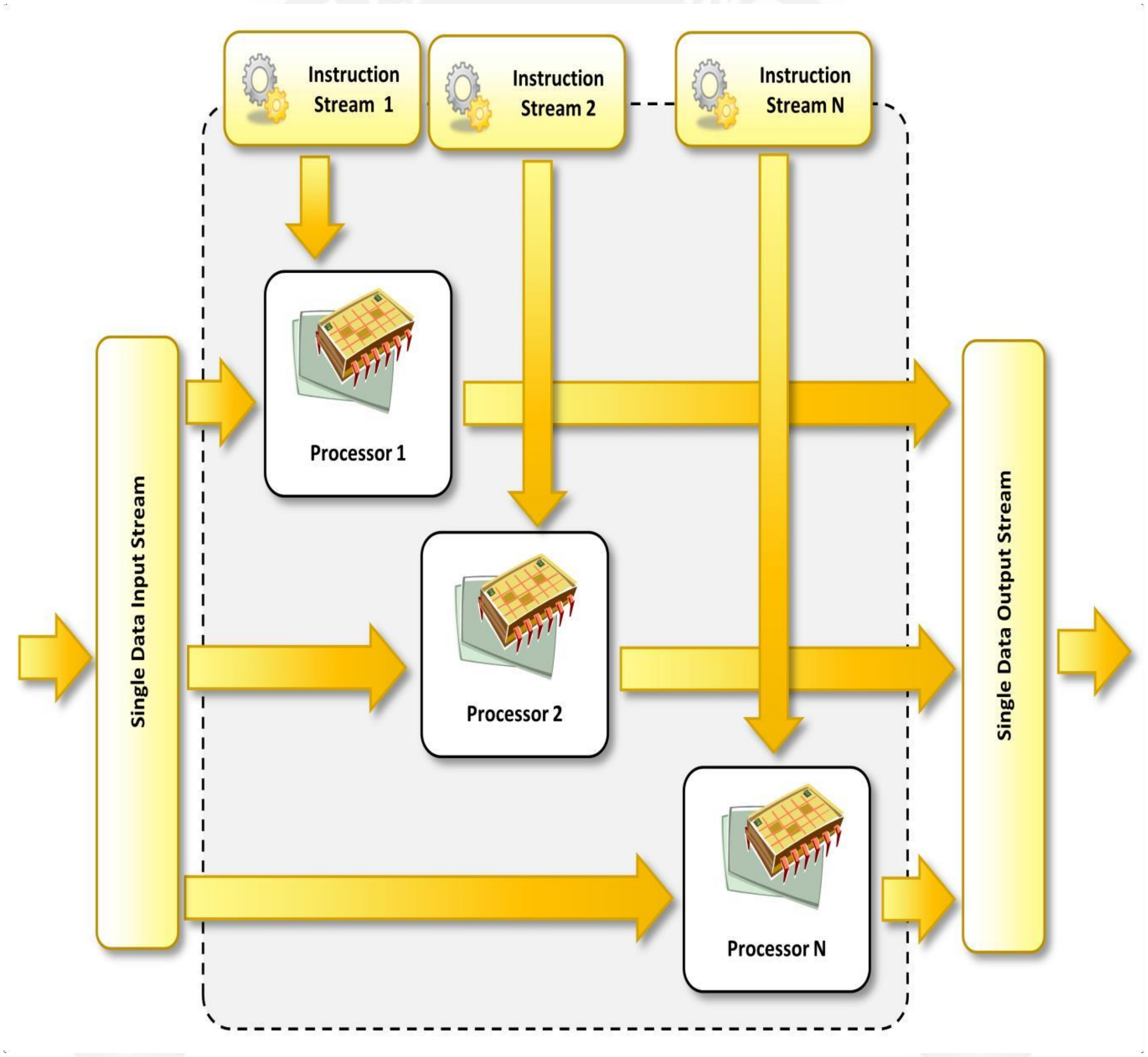
Dominant representative SIMD systems are Cray's Vector processing machine and Thinking Machines Cm*, and GPGPU accelerators



•

(iii) Multiple – Instruction , Single Data (MISD) systems

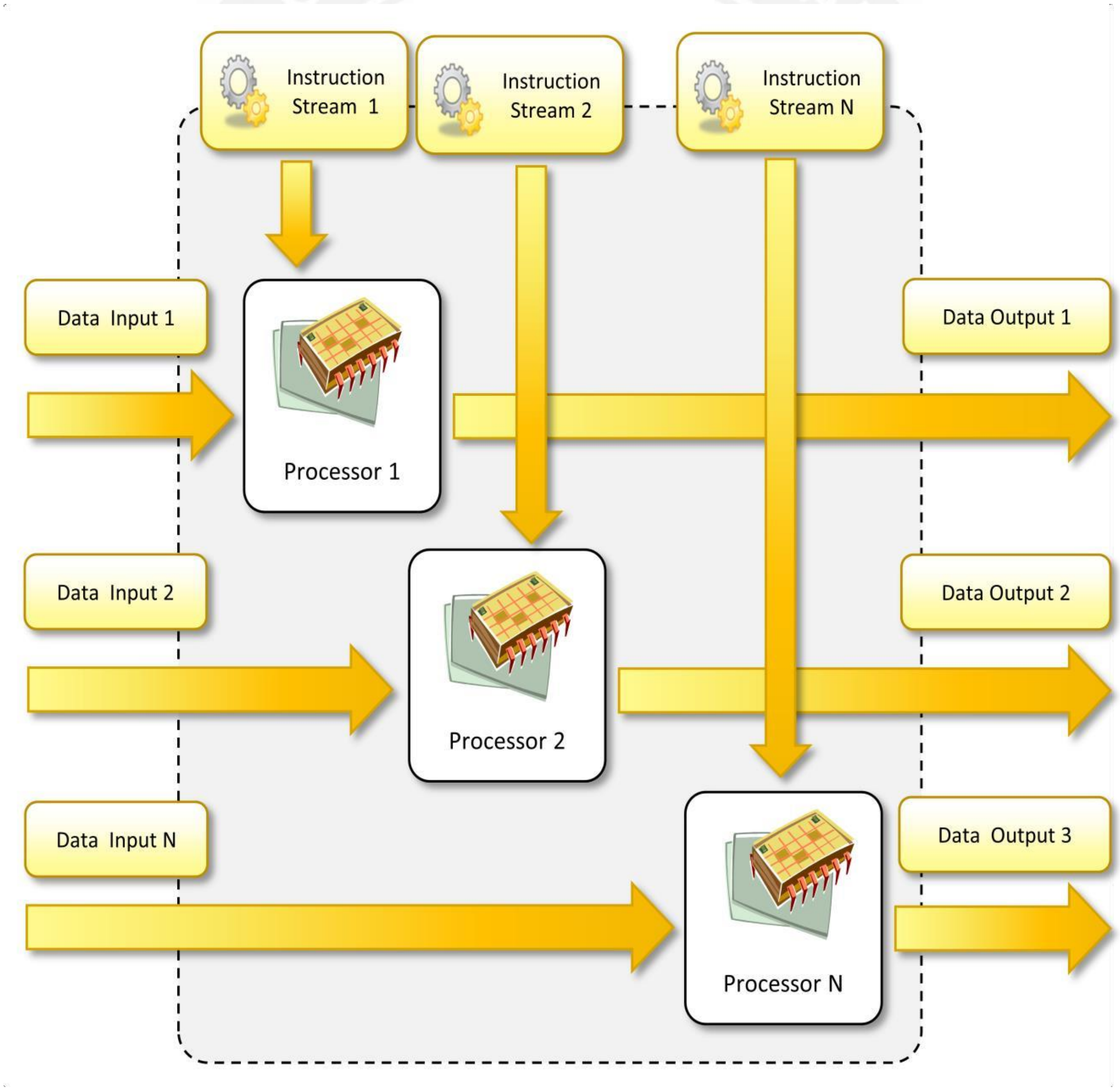
- MISD computing system is a multi processor machine capable of executing different instructions on different Pes all of them operating on the same data set.
- Machines built using MISD model are not useful in most of the applications.
- Few machines are built but none of them available commercially.
- This type of systems are more of an intellectual exercise than a practical configuration.



(iv) Multiple – Instruction , Multiple Data (MIMD) systems

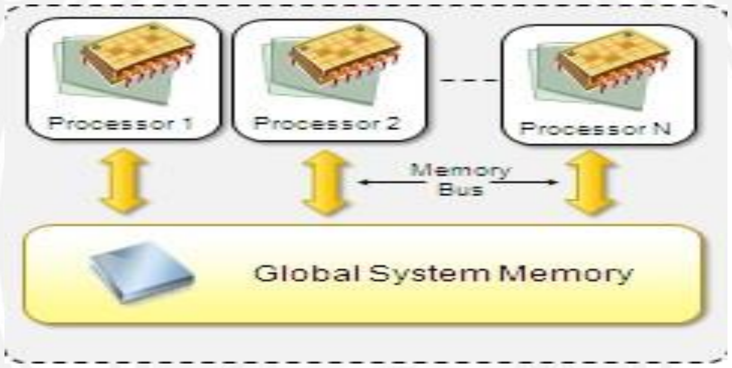
- MIMD computing system is a multi processor machine capable of executing multiple instructions on multiple data sets.
- Each PE in the MIMD model has separate instruction and data streams, hence machines built using this model are well suited to any kind of application.
- Unlike SIMD, MISD machine, PEs in MIMD machines work asynchronously,

MIMD machines are broadly categorized into shared-memory MIMD and distributed memory MIMD based on the way PEs are coupled to the main memory



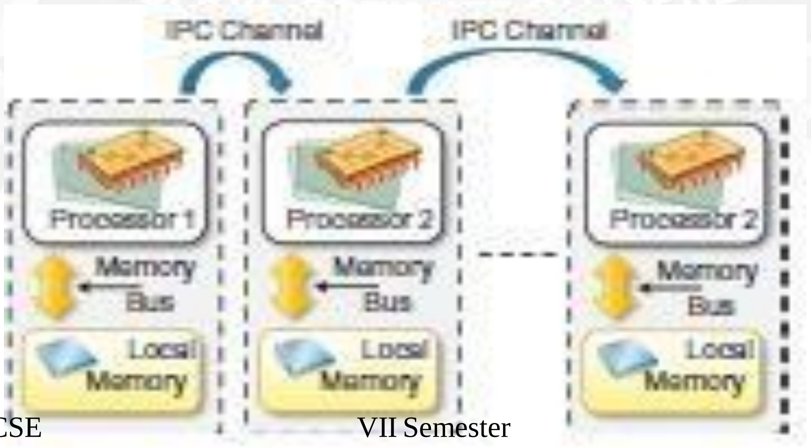
Shared Memory MIMD machines

- All the PEs are connected to a single global memory and they all have access to it.
- Systems based on this model are also called tightly coupled multi processor systems.
- The communication between PEs in this model takes place through the shared memory.
- Modification of the data stored in the global memory by one PE is visible to all other PEs.
- Dominant representative shared memory MIMD systems are silicon graphics machines and Sun/IBM SMP (Symmetric Multi-Processing).



Distributed Memory MIMD machines

- All PEs have a local memory. Systems based on this model are also called loosely coupled multi processor systems.
- The communication between PEs in this model takes place through the interconnection network, the inter process communication channel, or IPC.
- The network connecting PEs can be configured to tree, mesh, cube, and so on.
- Each PE operates asynchronously, and if communication/synchronization among tasks is necessary, they can do so by exchanging messages between them.



Shared Vs Distributed MIMD model

- The shared memory MIMD architecture is easier to program but is less tolerant to failures and harder to extend with respect to the distributed memory MIMD model.
- Failures, in a shared memory MIMD affect the entire system, whereas this is not the case of the distributed model, in which each of the PEs can be easily isolated.
- Moreover, shared memory MIMD architectures are less likely to scale because the addition of more PEs leads to memory contention.
- This is a situation that does not happen in the case of distributed memory, in which each PE has its own memory.

As a result, distributed memory MIMD architectures are most popular today

c.Approaches to Parallel Programming

- A sequential program is one that runs on a single processor and has a single line of control.
- To make many processors collectively work on a single program, the program must be divided into smaller independent chunks so that each processor can work on separate chunks of the problem.
- The program decomposed in this way is a parallel program.
- A wide variety of parallel programming approaches are available.
- The most prominent among them are the following.
- Data Parallelism
- Process Parallelism
- Farmer-and-worker model
- The above said three models are suitable for task-level parallelism. In the case of data level parallelism, the divide-and-conquer technique is used to split data into multiple sets,and each data set is processed on different PEs using the same instruction.
- This approach is highly suitable to processing on machines based on the SIMD model.
- In the case of Process Parallelism, a given operation has multiple (but distinct) activities that can be processed on multiple processors.
- In the case of Farmer-and-Worker model, a job distribution approach is used, one processor is configured as master and all other remaining PEs are designated as slaves,

the master assigns the jobs to slave PEs and, on completion, they inform the master, which in turn collects results.

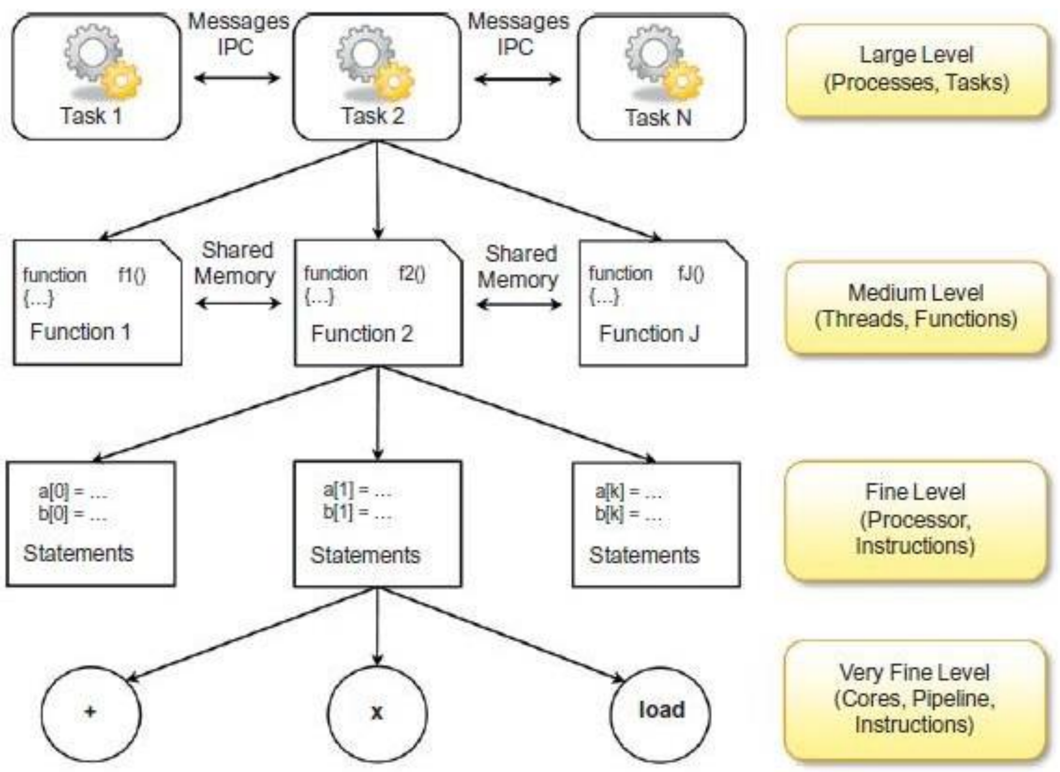
- These approaches can be utilized in different levels of parallelism.

d. Levels of Parallelism

- Levels of Parallelism are decided on the lumps of code (grain size) that can be a potential candidate of parallelism.
- The table shows the levels of parallelism.
- All these approaches have a common goal
  - To boost processor efficiency by hiding latency.
  - To conceal latency, there must be another thread ready to run whenever a lengthy operation occurs.
- The idea is to execute concurrently two or more single-threaded applications. Such as compiling, text formatting, database searching, and device simulation.

Grain Size	Code Item	Parallelized By
Large	Separate and heavy weight process	Programmer
Medium	Function or procedure	Programmer
Fine	Loop or instruction block	Parallelizing compiler
Very Fine	Instruction	Processor

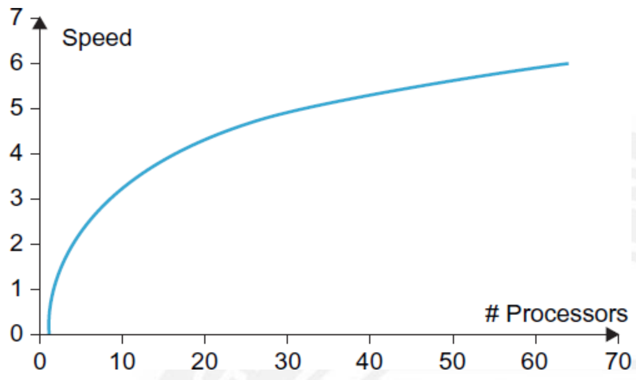
Levels of Parallelism

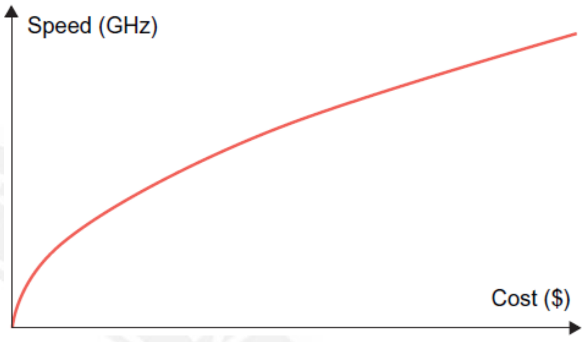


<u>e. Laws of Caution</u>

- Studying how much an application or a software system can gain from parallelism.
- In particular what need to keep in mind is that parallelism is used to perform multiple activities together so that the system can increase its throughput or its speed.
- But the relations that control the increment of speed are not linear.
- For example: for a given n processors, the user expects speed to be increase by in times. This is an ideal situation, but it rarely happens because of the communication overhead.
- Here two important guidelines to take into account.
  - Speed of computation is proportional to the square root of the system cost; they never increase linearly. Therefore, the faster a system becomes, the more expensive it is to increase its speed

– Speed by a parallel computer increases as the logarithm of the number of





processors (i.e. y=k*log(N)).

Number processors versus speed **Cost versus speed**

1.4.3 Elements of Distributed Computing

a.General concepts and definitions

- Distributed computing studies the models, architectures, and algorithms used for building and managing distributed systems.
- As general definition of the term distributed system, we use the one proposed by Tanenbaum
  - A distributed system is a collection of independent computers that appears to its users as a single coherent system.
- This definition is general enough to include various types of distributed computingsystems that are especially focused on unified usage and aggregation of distributedresources.
- Communications is another fundamental aspect of distributed computing. Sincedistributed systems are composed of more than one computer that collaborate together, it is necessary to provide some sort of data and information exchange between them, which generally occurs through the network.
  - A distributed system is one in which components located at networked computers communicate and coordinate their action only by passing messages.
- As specified in this definition, the components of a distributed system communicate with some sort of message passing. This is a term that encompasses several communication models.

b.Components of distributed System

- A distributed system is the result of the interaction of several components that traverse the entire computing stack from hardware to software.
- It emerges from the collaboration of several elements that- by working together- give users the illusion of a single coherent system.
- The figure provides an overview of the different layers that are involved in providing the services of a distributed system.

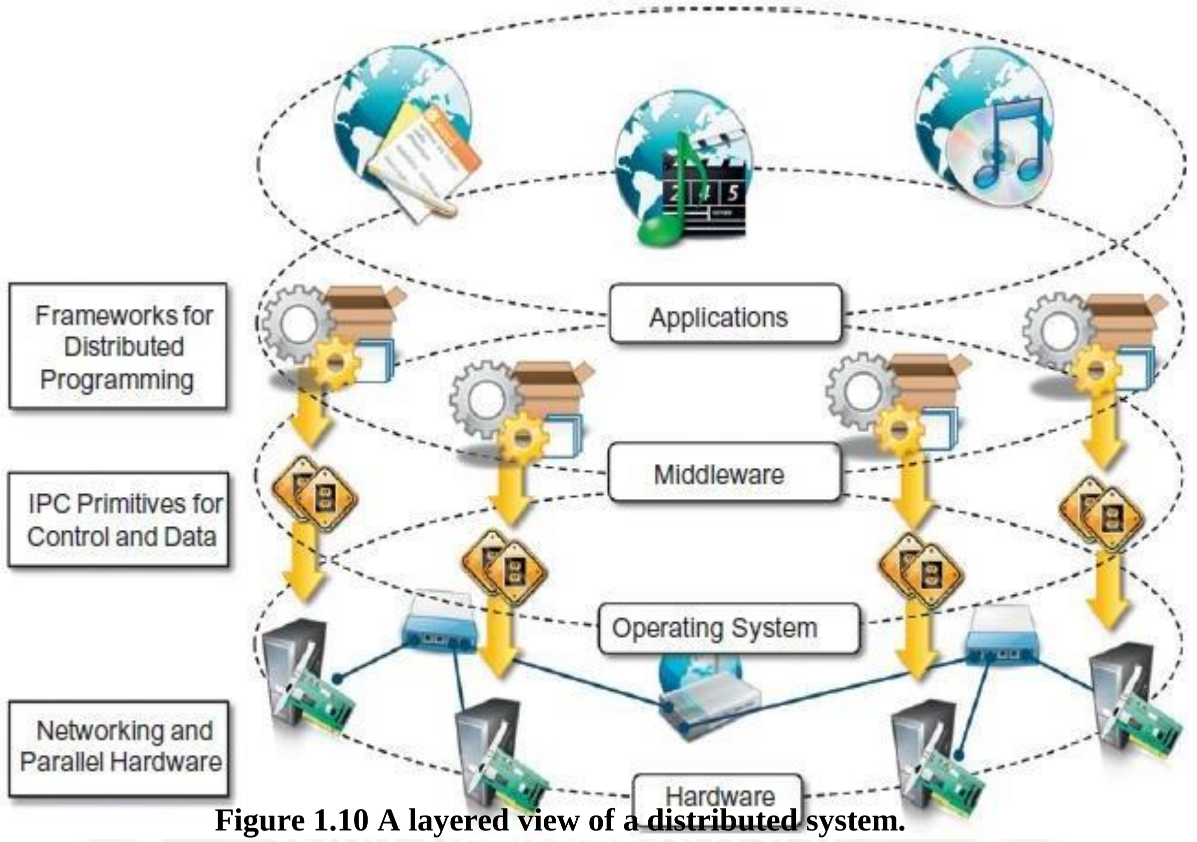

Frameworks for Distributed Programming
IPC Primitives for Control and Data
Networking and Parallel Hardware
Applications
Middleware
Operating System
Hardware

Figure 1.10 A layered view of a distributed system.

c.Architectural styles for distributed computing

- At the very bottom layer, computer and network hardware constitute the physical infrastructure; these components are directly managed by the operating system, which provides the basic services for inter process communication (IPC), process scheduling and management, and resource management in terms of file system and local devices.

- Taken together these two layers become the platform on top of which specialized software is deployed to turn a set of networked computers into a distributed system
- Although a distributed system comprises the interaction of several layers, the middleware layer is the one that enables distributed computing, because it provides a coherent and uniform runtime environment for applications.
- There are many different ways to organize the components that, taken together, constitute such an environment.
- The interactions among these components and their responsibilities give structure to the middleware and characterize its type or, in other words, define its architecture.
- Architectural styles aid in understanding the classifying the organization of the software systems in general and distributed computing in particular.
- The use of well-known standards at the operating system level and even more at the hardware and network levels allows easy harnessing of heterogeneous components and their organization into a coherent and uniform system.
- For example; network connectivity between different devices is controlled by standards, which allow them into interact seamlessly.
- Design patterns help in creating a common knowledge within the community of software engineers and developers as to how to structure the relevant of components within an application and understand the internal organization of software applications.
- Architectural styles do the same for the overall architecture of software systems.
- The architectural styles are classified into two major classes
    - Software Architectural styles : Relates to the logical organization of the software.
    - System Architectural styles: styles that describe the physical organization of distributed software systems in terms of their major components.

Software Architectural Styles

- Software architectural styles are based on the logical arrangement of software components.
- They are helpful because they provide an intuitive view of the whole system, despite its physical deployment.

- They also identify the main abstractions that are used to shape the components of the system and the expected interaction patterns between them.

Data Centered Architectures

- These architectures **identify the data as the fundamental element of the software system,** and access to shared data is the core characteristics of the data-centered architectures.
- Within the context of distributed and parallel computing systems, **integrity of data is overall goal for such systems.**
- The **repository architectural style** is the most relevant reference model in this category. It is characterized by two main components – the central data structure, which represents the current state of the system, and a collection of independent component, which operate on the central data.
- The ways in which the independent components interact with the central data structure can be very heterogeneous.
- In particular repository based architectures differentiate and specialize further into subcategories according to the choice of control discipline to apply for the shared data structure. Of particular interest are databases and blackboard systems.

Black board Architectural Style

- The black board architectural style is characterized by three main components:
  - Knowledge sources: These are entities that update the knowledge base that is maintained in the black board.
  - Blackboard: This represents the data structure that is shared among the knowledge sources and stores the knowledge base of the application.
  - Control: The control is the collection of triggers and procedures that govern the interaction with the blackboard and update the status of the knowledge base.

Data Flow Architectures

- Access to data is the core feature; data-flow styles explicitly incorporate the pattern of data-flow, since their design is determined by an orderly motion of data from component to component, which is the form of communication between them.
- Styles within this category differ in one of the following ways: how the control is exerted, the degree of concurrency among components, and the topology that describes the flow of data.

- **Batch Sequential:** The batch sequential style is characterized by an ordered sequence of separate programs executing one after the other. These programs are chained together by providing as input for the next program the output generated by the last program after its completion, which is most likely in the form of a file. This design was very popular in the mainframe era of computing and still finds applications today. For example, many distributed applications for scientific computing are defined by jobs expressed as sequence of programs that, for example, pre-filter, analyze, and post process data. It is very common to compose these phases using the batch sequential style.
- **Pipe-and-Filter Style:** It is a variation of the previous style for expressing the activity of a software system as sequence of data transformations. Each component of the processing chain is called a filter, and the connection between one filter and the next is represented by a data stream.

Virtual Machine architectures

- The virtual machine class of architectural styles is characterized by the presence of an abstract execution environment (generally referred as a virtual machine) that simulates features that are not available in the hardware or software.
- Applications and systems are implemented on top of this layer and become portable over different hardware and software environments.
- The general interaction flow for systems implementing this pattern is – the program (or the application) defines its operations and state in an abstract format, which is interpreted by the virtual machine engine. The interpretation of a program constitutes its execution. Itis quite common in this scenario that the engine maintains an internal representation ofthe program state.
- Popular examples within this category are rule based systems, interpreters, and command language processors.
- **Rule-Based Style:**
  - This architecture is characterized by representing the abstract execution environment as an inference engine. Programs are expressed in the form of rules or predicates that hold true. The input data for applications is generally represented by a set of assertions or facts that the inference engine uses to activaterules or to apply predicates, thus transforming data. The examples of rule-based systems can be found in the networking domain: Network Intrusion Detection

Systems (NIDS) often rely on a set of rules to identify abnormal behaviors connected to possible intrusion in computing systems.

- Interpreter Style: The presence of engine to interpret the style.

Call and return architectures

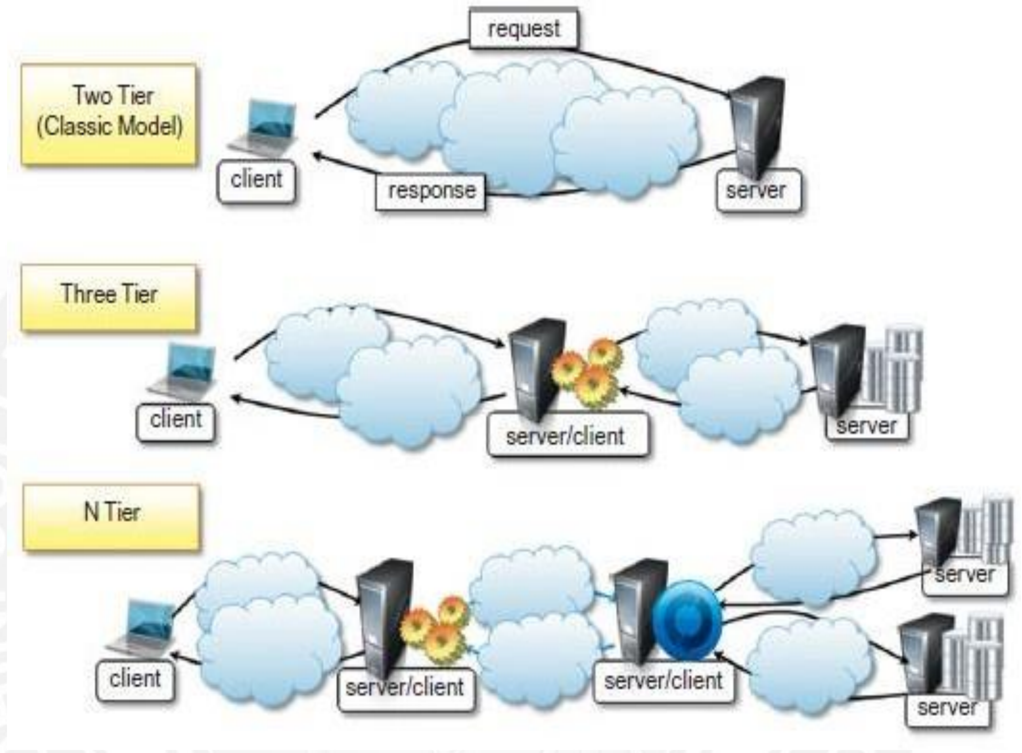
- This identifies all systems that are organized into components mostly connected together by method calls.
- The activity of systems modeled in this way is characterized by a chain of method calls whose overall execution and composition identify the execution one or more operations.
- There are three categories in this
  - Top down Style : developed with imperative programming
  - Object Oriented Style: Object programming models
  - Layered Style: provides the implementation in different levels of abstraction of the system.

System Architectural Styles

- System architectural styles cover the physical organization of components and processes over a distributed infrastructure.
- Two fundamental reference style
  - Client / Server
- Peer- to -Peer
- The information and the services of interest can be centralized and accessed through a single access point: the server.
- Multiple clients are interested in such services and the server must be appropriately designed to efficiently serve requests coming from different clients.

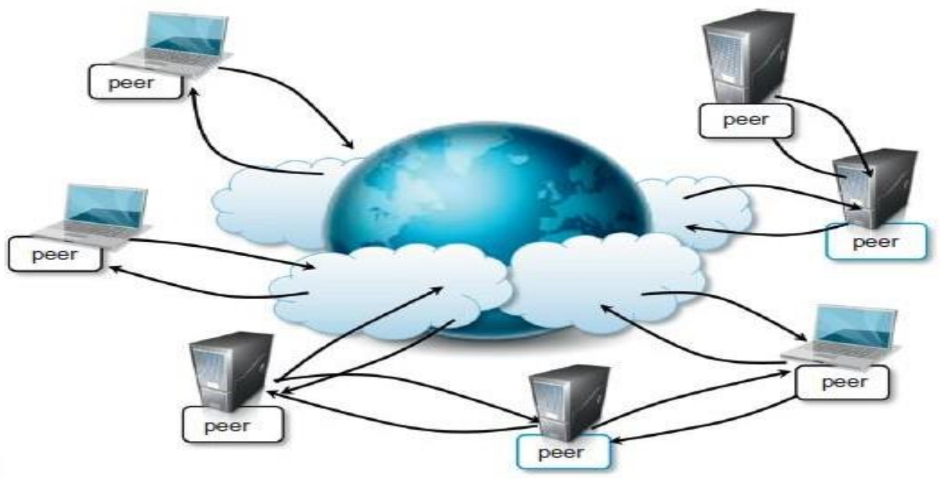
Client / Server architectural Styles

-
-
-
-
-



- Symmetric architectures in which all the components, called peers, play the same role and incorporate both client and server capabilities of the client/server model.
- More precisely, each peer acts as a server when it processes requests from other peers and as a client when it issues requests to other peers.

Peer-to-Peer architectural Style


peer
peer
peer
peer
peer
peer
peer
peer

d.Models for Inter process Communication

- Distributed systems are composed of a collection of concurrent processes interacting with each other by means of a network connection.
- IPC is a fundamental aspect of distributed systems design and implementation.
- IPC is used to either exchange data and information or coordinate the activity of processes.
- IPC is what ties together the different components of a distributed system, thus making them act as a single system.
- There are several different models in which processes can interact with each other – these maps to different abstractions for IPC.
- Among the most relevant that we can mention are shared memory, remote procedure call (RPC), and message passing.
- At lower level, IPC is realized through the fundamental tools of network programming.
- Sockets are the most popular IPC primitive for implementing communication channels between distributed processes.

Message-based communication

- The abstraction of message has played an important role in the evolution of the model and technologies enabling distributed computing.
- The definition of distributed computing – is the one in which components located at networked computers communicate and coordinate their actions only by passing messages. The term messages, in this case, identify any discrete amount of information that is passed from one entity to another. It encompasses any form of data representation that is limited in size and time, whereas this is an invocation to a remote procedure or a serialized object instance or a generic message.
- The term message-based communication model can be used to refer to any model for IPC.
- Several distributed programming paradigms eventually use message-basedcommunication despite the abstractions that are presented to developers for programmingthe interactions of distributed components.
- Here are some of the most popular and important:

Message Passing: This paradigm introduces the concept of a message as the main abstraction of the model. The entities exchanging information explicitly encode in the formof a message the data to be exchanged. The structure and the content of a message vary according to the model. Examples of this model are the Message-Passing-Interface (MPI)and openMP.

- **Remote Procedure Call (RPC):** This paradigm extends the concept of procedure call beyond the boundaries of a single process, thus triggering the execution of code in remote processes.
- **Distributed Objects:** This is an implementation of the RPC model for the object-oriented paradigm and contextualizes this feature for the remote invocation of methods exposed by objects. Examples of distributed object infrastructures are Common Object Request Broker Architecture (CORBA), Component Object Model (COM, DCOM, and COM+), Java Remote Method Invocation (RMI), and .NET Remoting.
- **Distributed agents and active Objects:** Programming paradigms based on agents and active objects involve by definition the presence of instances, whether they are agents of objects, despite the existence of requests.
- **Web Service**: An implementation of the RPC concept over HTTP; thus allowing the interaction of components that are developed with different technologies. A Web service is exposed as a remote object hosted on a Web Server, and method invocation are transformed in HTTP requests, using specific protocols such as Simple Object Access Protocol (SOAP) or Representational State Transfer (REST).

e. Technologies for distributed computing

- Remote Procedure Call (RPC)
  - RPC is the fundamental abstraction enabling the execution procedures on clients' request.
  - RPC allows extending the concept of a procedure call beyond the boundaries of a process and a single memory address space.
  - The called procedure and calling procedure may be on the same system or they may be on different systems..
- Distributed Object Frameworks

  - Extend object-oriented programming systems by allowing objects to be distributed across a heterogeneous network and provide facilities so that they can be coherently act as though they were in the same address space.

Web Services

- Web Services are the prominent technology for implementing SOA systems and applications.
- They leverage Internet technologies and standards for building distributed systems.
- Several aspects make Web Services the technology of choice for SOA.
- First, they allow for interoperability across different platforms and programming languages.
- Second, they are based on well-known and vendor-independent standards such as HTTP, SOAP, and WSDL.
- Third, they provide an intuitive and simple way to connect heterogeneous softwaresystems, enabling quick composition of services in distributed environment.