Principles of Parallel & Distributed Computing

Dr. Subrajeet Mohapatra
Assistant Professor
Department of Computer Science & Engineering,
BIT Mesra, Ranchi
Jharkhand

Copyright Information

- The course materials is intended only for **instructional purpose** of students of BIT Mesra
- Students can take notes and make copies of course materials only for individual use
- Enrolled students should not reproduce, distribute or display (post/upload) lecture notes or recordings or course materials in any other way without my express written consent
- The teacher in-charge for this course material will not be responsible for violation of above policies by the students
- Students violating the above policies may be subject to student conduct proceedings

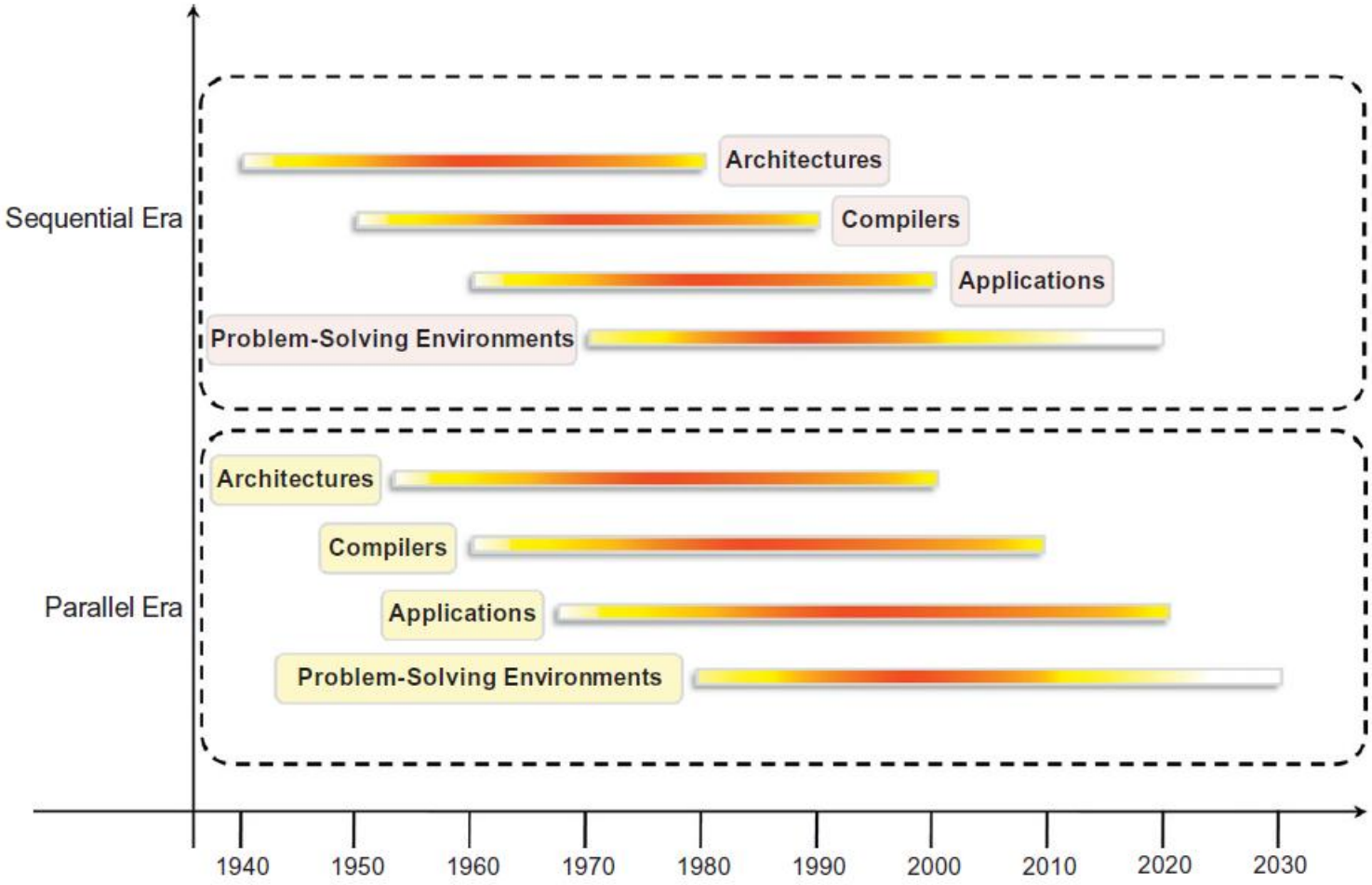
Disclaimer & Fair Dealing Statement

- The author of this course material will not be held personally responsible, nor liable for any damages, actual or consequential, for any posts by third parties which may violate any law.
- This course material may contain copyrighted material the use of which has not always been specifically authorized by the copyright owner
- Use of materials in the presentation constitutes a ‘fair dealing of any such copyrighted material
- The material contained in this presentation is used without profit for research and educational purposes only

Introduction

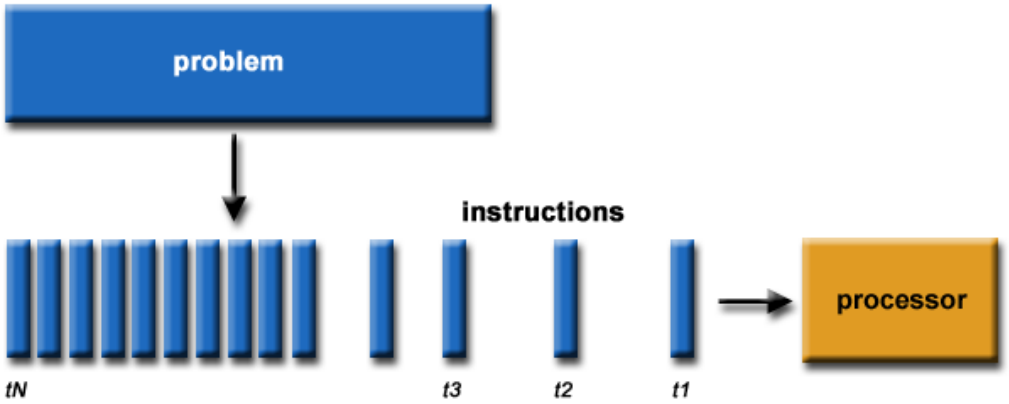
- Cloud computing is a new technological trend that supports better utilization of IT infrastructures, services, and applications.
- Adopts a service delivery model based on a **pay-per-use** approach, in which users do not own infrastructure, platform, or applications but use them for the time they need them.
- IT assets are **owned and maintained** by **service providers** who make them accessible through the Internet.
- Parallel and distributed computing serve as foundations for building cloud computing systems and applications.

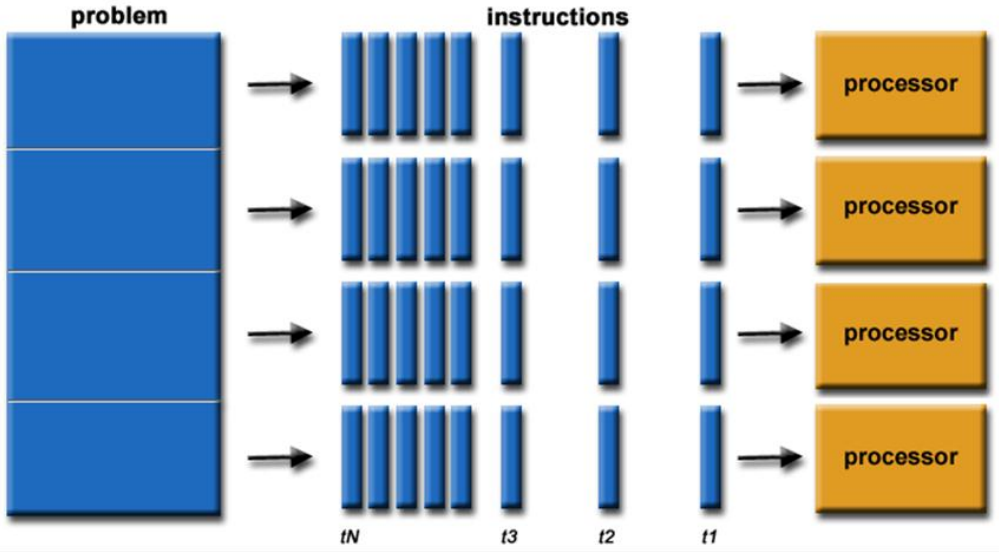
Eras of Computing



Serial vs. Parallel Computing

Serial Computing





Parallel Computing

Parallel & Distributed Computing

- Parallel and Distributed computing are often used interchangeably even though they mean slightly different things.
- They are based on systems concepts, such as concurrency, mutual exclusion, consistency in state/memory manipulation, message-passing, and shared-memory models.
- The term parallel implies a **tightly coupled system**, whereas distributed refers to a wider class of system, **including those that are tightly coupled**.

Parallel Computing

- Refers to a computing model in which the computation is divided among several processors **sharing the same memory**.
- The architecture of a 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.
- Originally only those architectures were considered as parallel systems that featured multiple processors **sharing the same physical memory** (Considered as a single computer).
- Over time, 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.

Distributed Computing

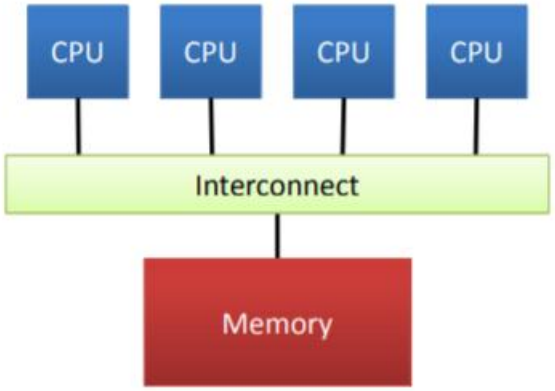
- Distributed computing (DC) encompasses any architecture or system that allows the computation to be broken down into units and executed concurrently.
- The program execution can be on different computing elements, i.e. processors on different nodes, processors on the same computer, or cores within the same processor.
- DC includes a wider range of systems and applications than parallel computing and is often considered a more general term.
- Locations of the computing elements are usually not the same and such elements might be **heterogeneous** in terms of hardware and software features.
- DC includes a **wider range of systems** and applications than parallel computing and is often considered a more **general term**.

Parallel vs. Distributed Computing

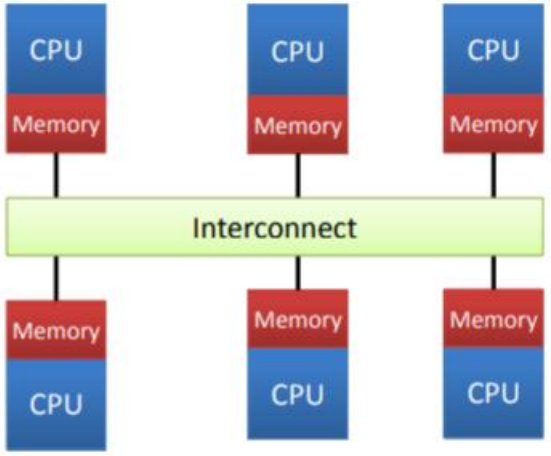
- **Parallel Computing** : All processors may have access to a shared memory to exchange information between processors.
- **Distributed computing** : Each processor has its own private memory (distributed memory) and information is exchanged by passing messages between the processors.

Parallel Systems	Distributed Systems
Tight coupling	Loose coupling
Physical proximity	Server room to Global
Homogeneous	Heterogeneous
Threads & MPI	RPC, Web Services, REST

Shared vs. Distributed Memory



Shared Memory



Distributed Memory

Elements of Parallel Computing

Necessity of Parallel Computing

- Computational requirements are ever increasing
- Wastage of hardware resources in serial processing
- Sequential architectures are reaching their physical limits
- Processing speed is constrained by the speed of light
- Density of transistors packaged in a processor is constrained by thermodynamic limitations.

Possible Solutions

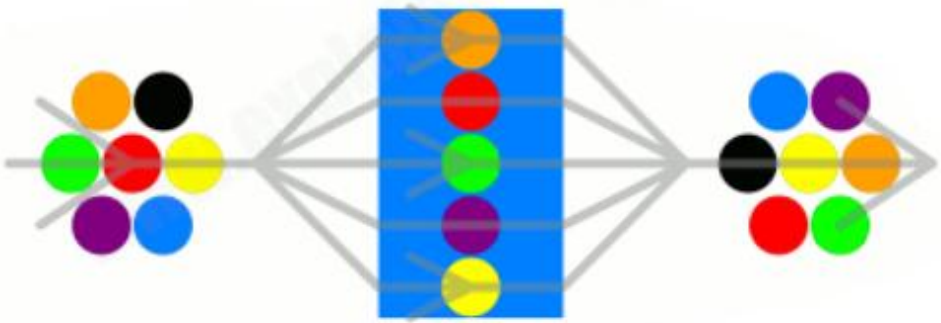
- Connecting multiple processors working in mutual coordination.
- Developing techniques, architectures, and systems for performing multiple activities in parallel

What is 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.
- Each task is divided into multiple subtasks using a divide-and-conquer technique, and each subtask is processed on a different CPU.
- Programming on a multiprocessor system using the divide-and-conquer technique is called parallel programming

Serial vs. Parallel Computing

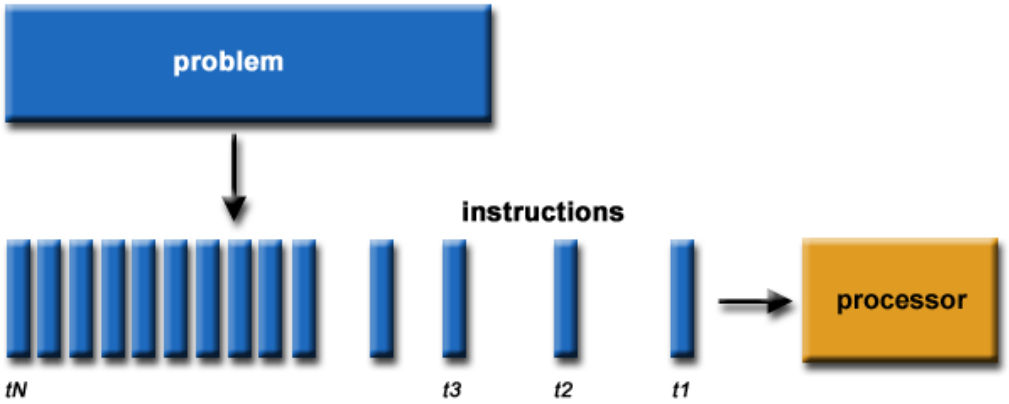
Serial Computing

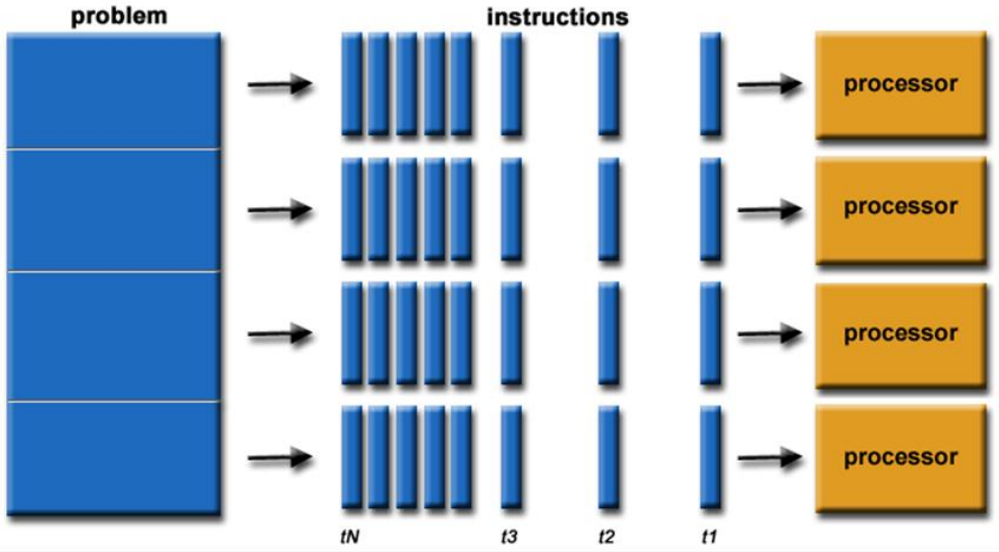


Parallel Computing

Serial vs. Parallel Computing

Serial Computing





Parallel Computing

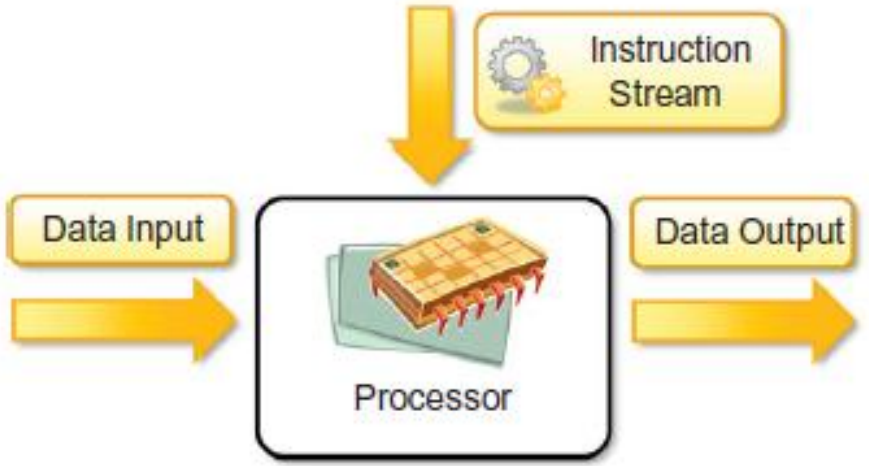
H/W Architectures for Parallel Processing

- Based on the number of instruction 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

H/W Architectures for Parallel Processing

Single-instruction, single-data (SISD) systems

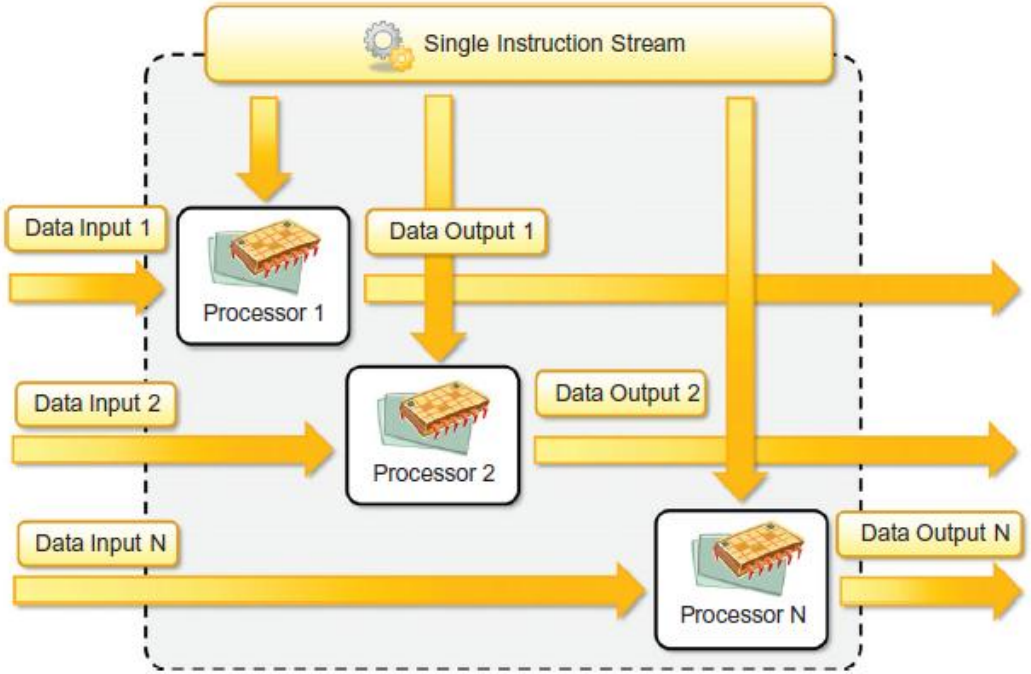
- A uniprocessor machine capable of executing a single instruction, which operates on a single data stream.
- Machine instructions are processed sequentially
- Instructions and data to be processed have to be stored in primary memory



H/W Architectures for Parallel Processing

Single-instruction, multiple-data (SIMD) systems

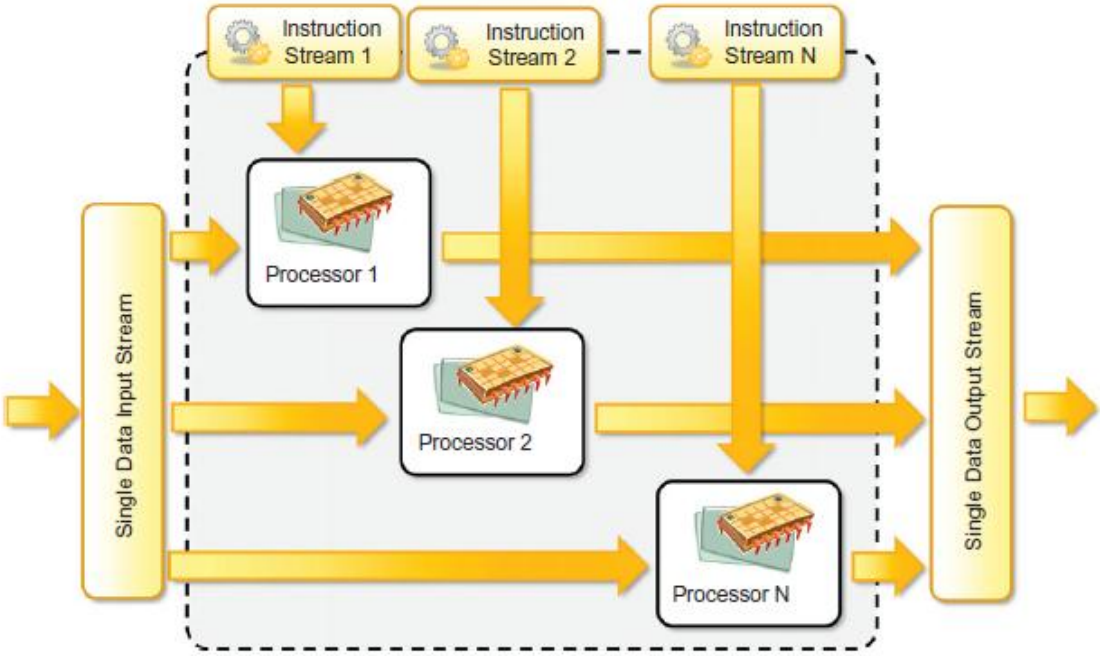
- A multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams.
- SIMD based machines are well suited for scientific computing.



H/W Architectures for Parallel Processing

Multiple-instruction, single-data (MISD) systems

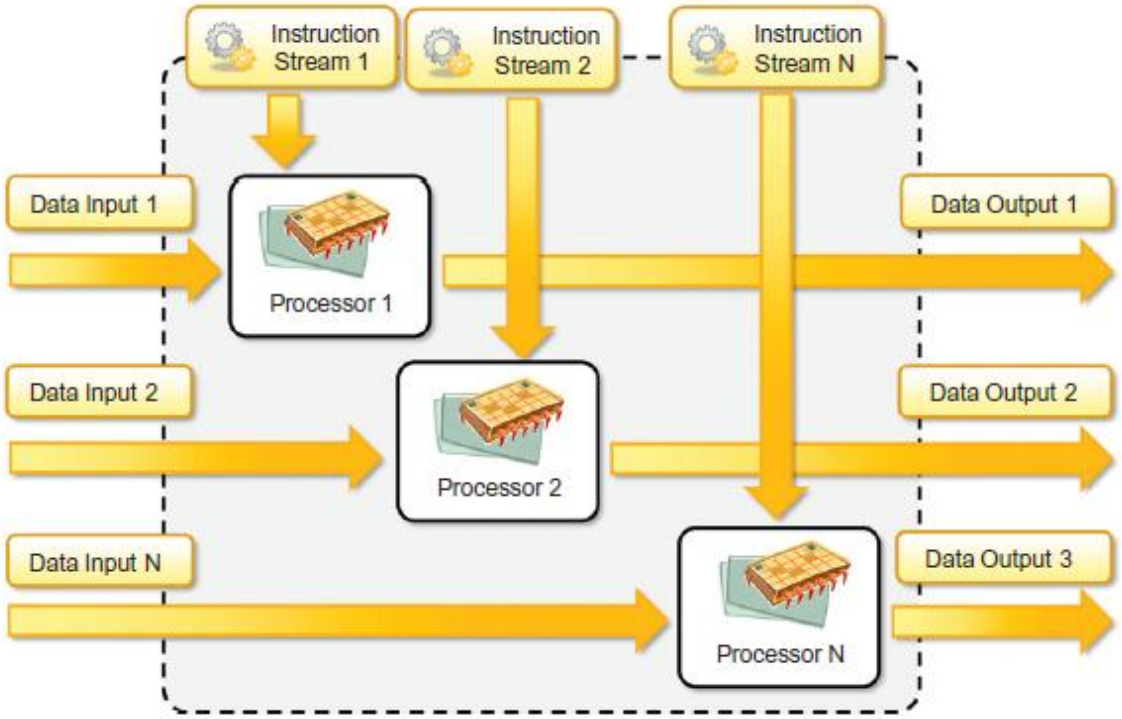
- A multiprocessor machine capable of executing different instructions on different processing elements but all of them operating on the same data set



H/W Architectures for Parallel Processing

Multiple-instruction, Multiple-data (MIMD) systems

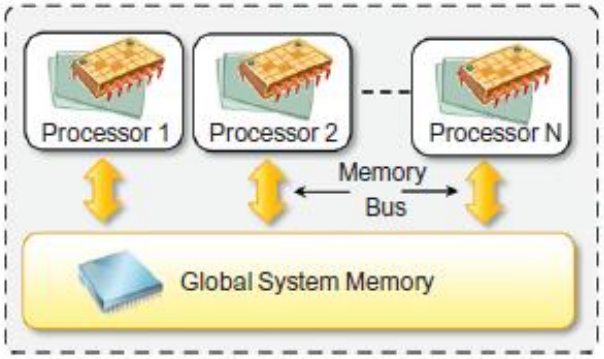
- A multiprocessor machine capable of executing multiple instructions on multiple data set



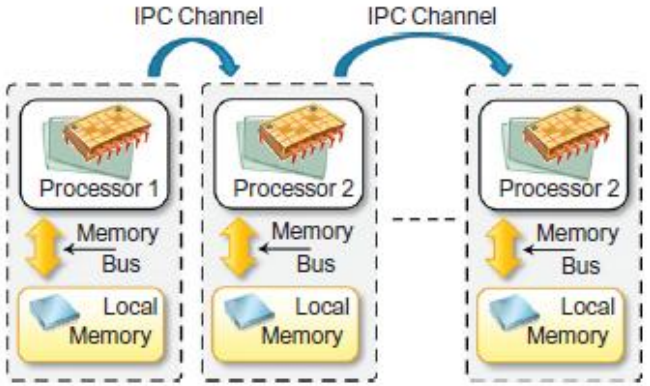
H/W Architectures for Parallel Processing

Classification of MIMD systems

- A multiprocessor machine capable of executing multiple instructions on multiple data set
- Processing elements in MIMD machines work asynchronously
- MIMD machines are broadly categorized based on the way PEs are coupled to the main memory i.e. Shared-memory MIMD and Distributed-memory MIMD



Shared-memory MIMD



Distributed-memory MIMD

Approaches to Parallel Programming

Parallel Program :

- The process of decomposing the program into smaller independent chunks of code is called a parallel program.
- Parallel program facilitates working collectively of many processors.

Different Parallel Programming Approaches

- **Data Parallelism :** 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.
- **Process Parallelism :** Operations with multiple (but distinct) activities can be processed on multiple processors.
- **Farmer-and-worker model :** Based on job distribution approach (One processor is configured as **master** and all other remaining PEs are designated as **slaves**)

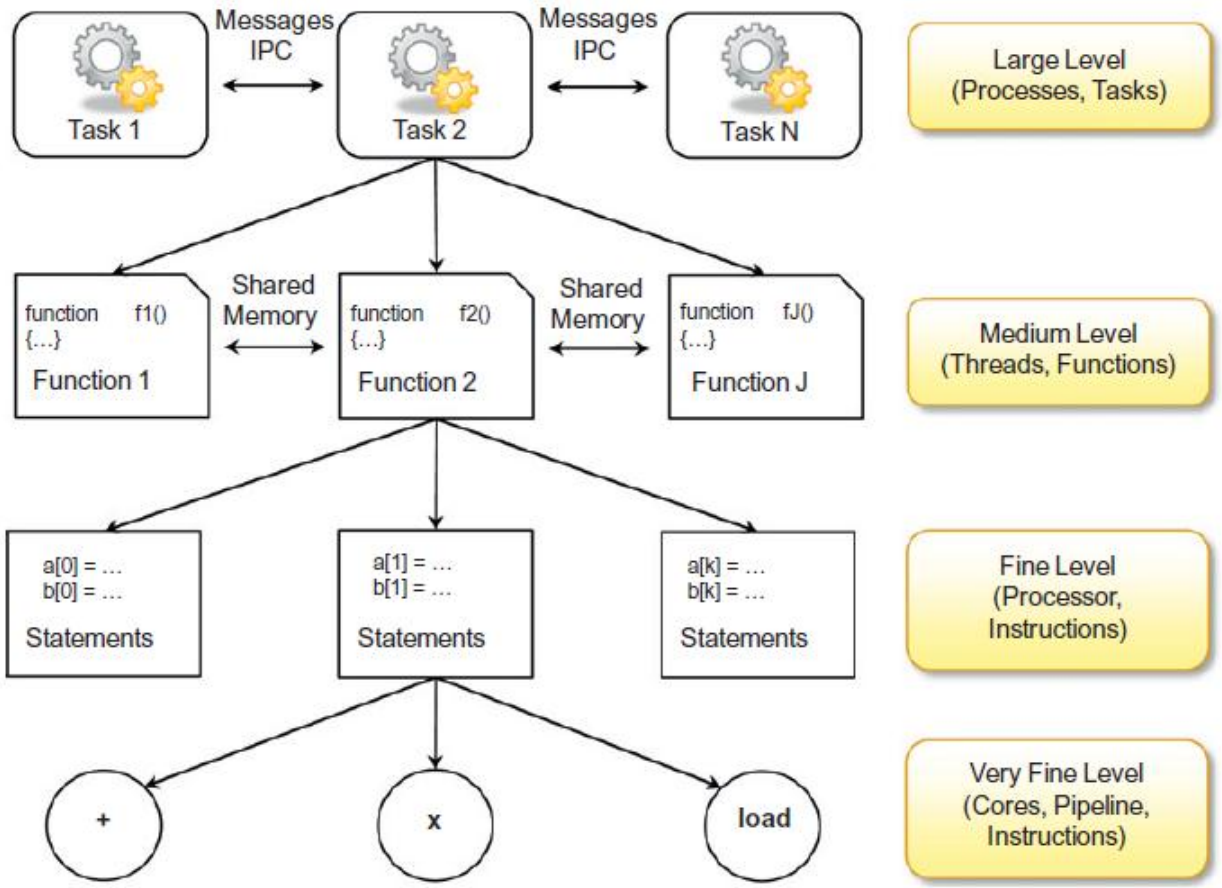
Levels of Parallelism

- Parallelism within an application are based on the lumps of code (grain size) a potential candidate for parallelism.
- Enables executing concurrently two or more single-threaded applications.
- Intends to boost processor efficiency by hiding latency.

Levels of Parallelism

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

Levels of Parallelism



Elements of Distributed Computing

General concepts and definitions

- Distributed computing studies the models, architectures, and algorithms used for building and managing distributed systems.

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

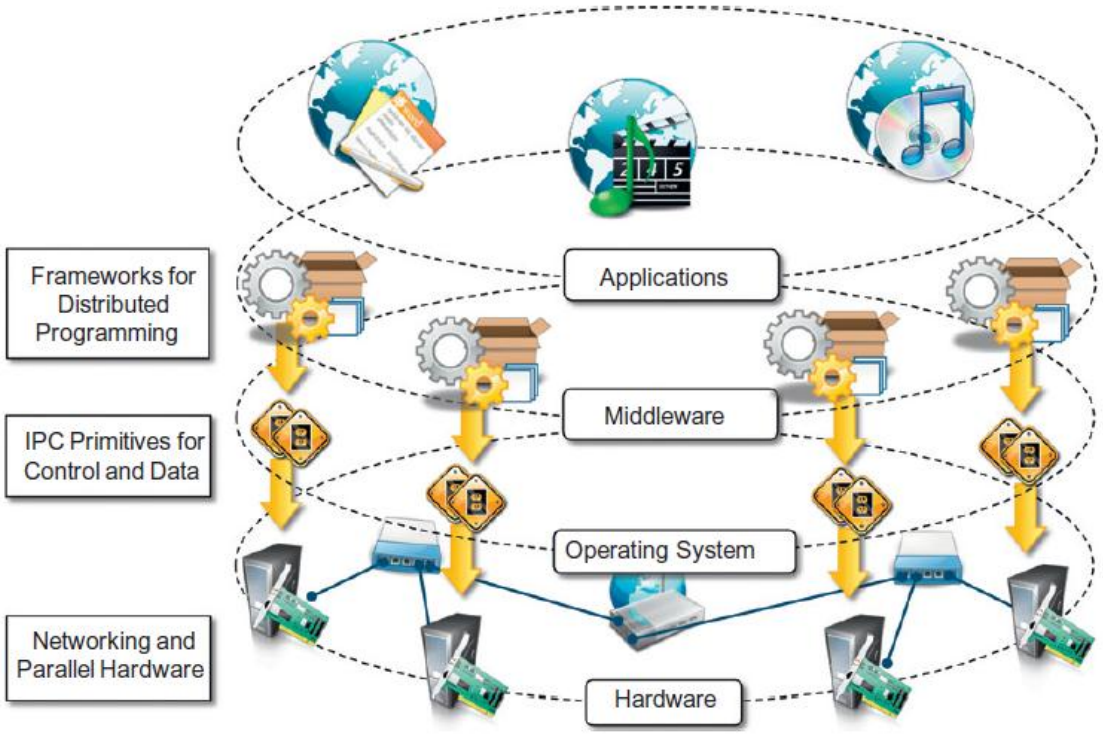
  "A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages."

- The components of a distributed system communicate with some sort of ***message passing***.
- Message passing encompasses several communication models.

Components of a Distributed System

- A distributed system is the result of the interaction of several components that traverse the entire computing stack from hardware to software.

A layered view of Distributed System



Components of a Distributed System

- **Bottom Layer :**
  - Physical Infrastructure
  - Operating System : IPC, Process Scheduling, File system management
- **Middleware Layer :**
  - Leverages services to build a uniform environment for the development and deployment of distributed applications.
  - Supports programming paradigms for distributed systems
  - Provides a uniform interface to developers (independent from the underlying OS and hiding the heterogeneities of the bottom layers)
- **Application Layer :**
  - Provides GUI's designed and developed to use the middleware.

Components of a Distributed Computing

- **Bottom Layer :**
  - Make up the physical infrastructure of one or more datacenters, where racks of servers are deployed and connected together through high-speed connectivity.
  - OS manages this infrastructure (machine and network management
- **Middleware Layer :**
  - Core logic is implemented in the middleware
  - Provides a customizable runtime environment for applications
- **Application Layer :**
  - Services are offered through Web 2.0-compliant interfaces.
  - Facilities catered ranges from virtual infrastructure building and deployment to application development and runtime environments.

Architectural Styles of Distributed Computing

- *"Architectural styles helps in understanding the interactions among different components (SW & HW) and their responsibilities in a distributed computing environment (*across multiple machines.*)".*
- *"Architectural styles are mainly used to determine the vocabulary of components and connectors that are used as instances of the style together with a set of constraints on how they can be combined".*
- Architectural styles for distributed systems can be categorized into two major classes :
  - Software architectural styles --Logical organization of the software
  - System architectural styles--Physical organization of distributed software

Architectural Styles of Distributed Computing

Components & Connectors

- Component represents a unit of software that encapsulates a function or a feature of the system.
- Examples of components can be programs, objects, processes, pipes, and filters.
- Connector is a communication mechanism that allows cooperation and coordination among different components.
- They are not encapsulated in a single entity and are implemented in a distributed manner over many system components.

Software Architectural Styles

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

Software Architectural Styles

Category	Most Common Architectural Styles
Data-centered	Repository Blackboard
Data flow	Pipe and filter Batch sequential
Virtual machine	Rule-based system Interpreter
Call and return	Main program and subroutine call/top-down systems Object-oriented systems Layered systems
Independent components	Communicating processes Event systems

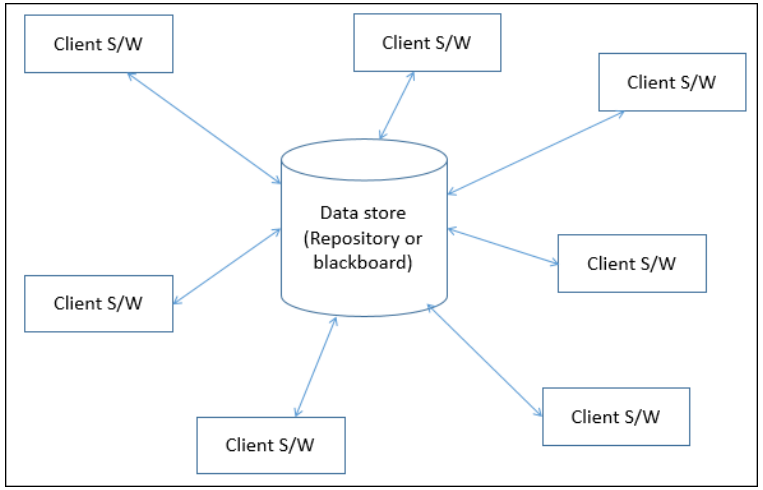
Software Architectural Styles

Data Centered Architecture :

- Data is centralized and accessed frequently by other components, which modify data.
- Achieving integrality of data is most important
- Different components communicate through shared data repositories
- Example: **Database** architecture and **Web** architecture

Types of Components

- A **central data** structure or data store
- A **data accessor** or collection of different components operate on the central data store and perform computations, and put back the results.



Software Architectural Styles

Data Centered Architecture :

- Communication between the data accessors is only through the data store.
- The data is the only means of communication among clients.
- Flow of control differentiates the architecture into two categories :-
  - **Repository architecture Style :** The data store is passive and the clients (software components or agents) are active, which control the flow logic.
  - **Blackboard architecture Style :** The data store is active and the clients are passive, and the logical flow is determined by the current data status of the data store.

Software Architectural Styles

Data Flow Architecture

- The whole software system is viewed as a series of transformations on consecutive pieces of input data, where data and operations are independent of each other.
- Data enters into the system and then flows through the modules one at a time until they are assigned to some final destination (output or a data store).
- Connections between the components may be implemented as I/O stream, I/O buffers, piped, or other types of connections.
- Suitable for applications involving a well-defined series of independent data transformations (computations) on orderly defined input and output.

Software Architectural Styles

Data Flow Architecture

- Data Flow Architecture approaches can be classified based on execution of sequences between modules
  - Batch Sequential
  - Pipe and Filter
  - Process Control

Software Architectural Styles

Batch Sequential :

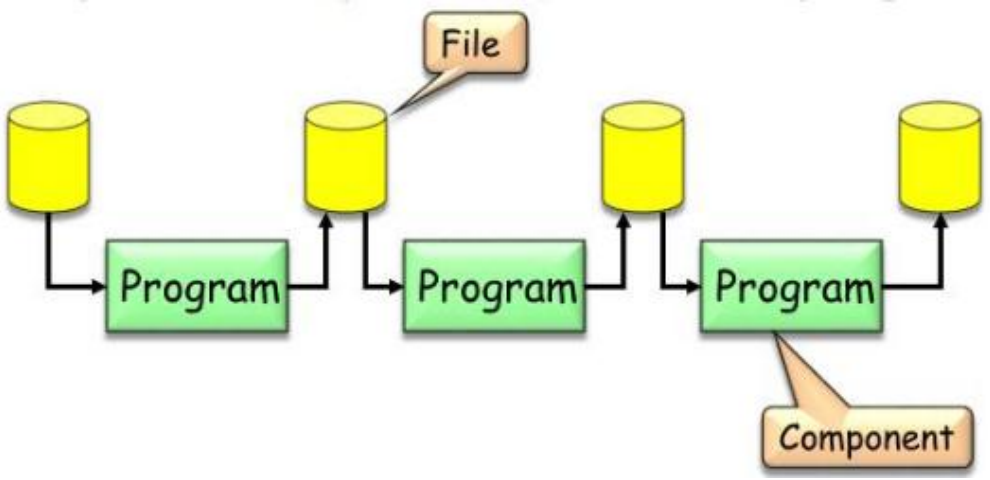
- A sequential data processing model
- Separate programs are executed in order and the data is passed as an aggregate from one program to the next.
- Provides simpler divisions on subsystems and each can be an independent program working on input data and produces output data.

Features

- Components are independent programs
- Connectors are media (typically files)
- Each step runs to completion before next step begins

Disadvantages

- Provides high latency and low throughput.
- Does not provide concurrency and interactive interface.



Software Architectural Styles

Pipe and Filter:

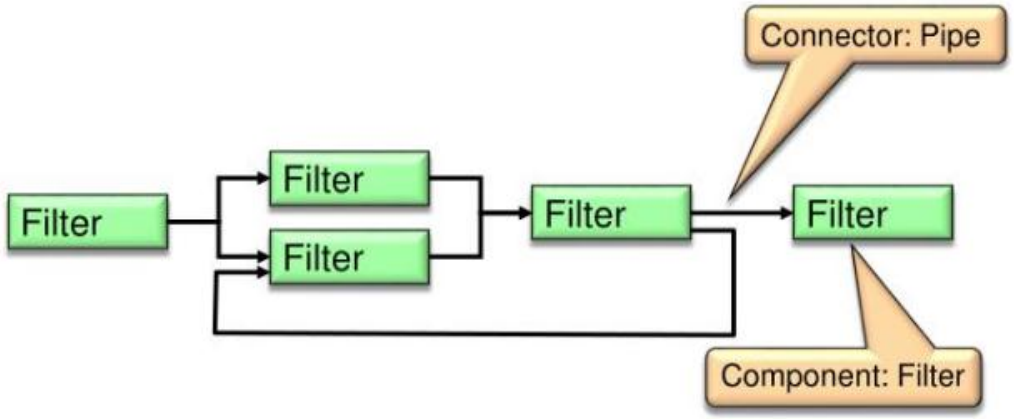
- A variation of the batch sequential style for expressing the activity of a software system as sequence of data transformations.
- Filter is an independent entity which perform transformation on data and process the input they receive.
- Pipes serve as connectors for the stream of data being transformed, each connected to the next component in the pipeline.

Features

- Ensures loose and flexible coupling of components, filters
- Supports parallel processing
- Each filter can be called and used over and over again

Disadvantages

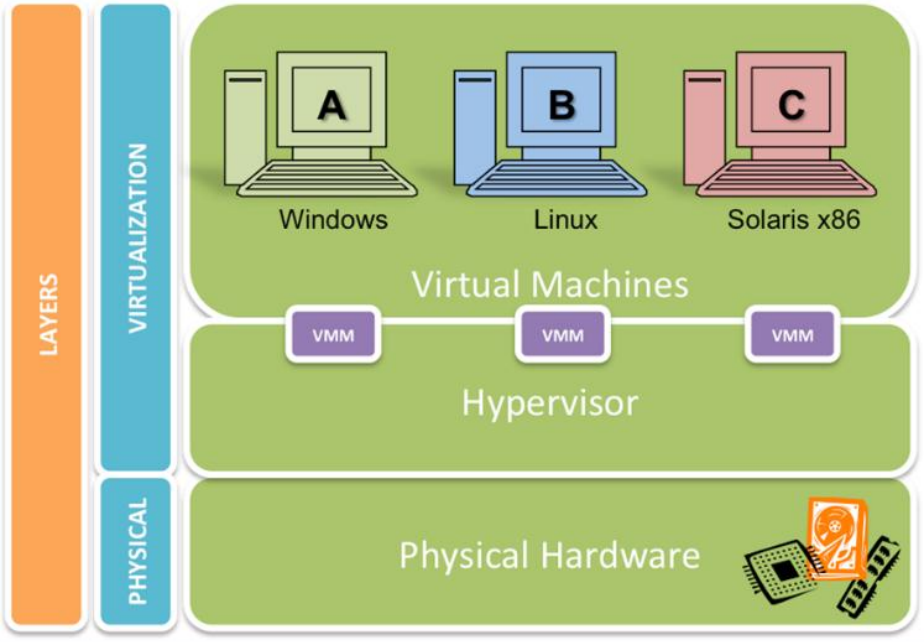
- Reduced performance due to excessive computational overheads.
- Unsuitable for an interactive system
- May not be appropriate for long-running computations.

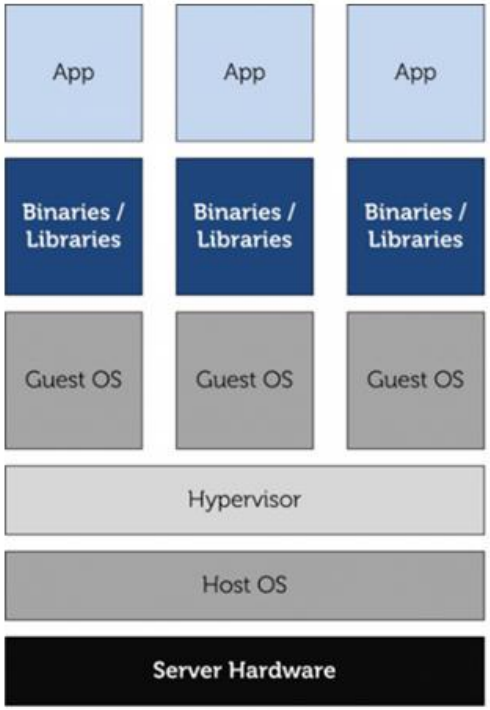


Software Architectural Styles

Virtual Machine Architectures

- Characterized by the presence of an abstract execution environment (VM)
- Simulates features that are not available in the H/W or S/W
- Applications are implemented on top of this virtualization layer





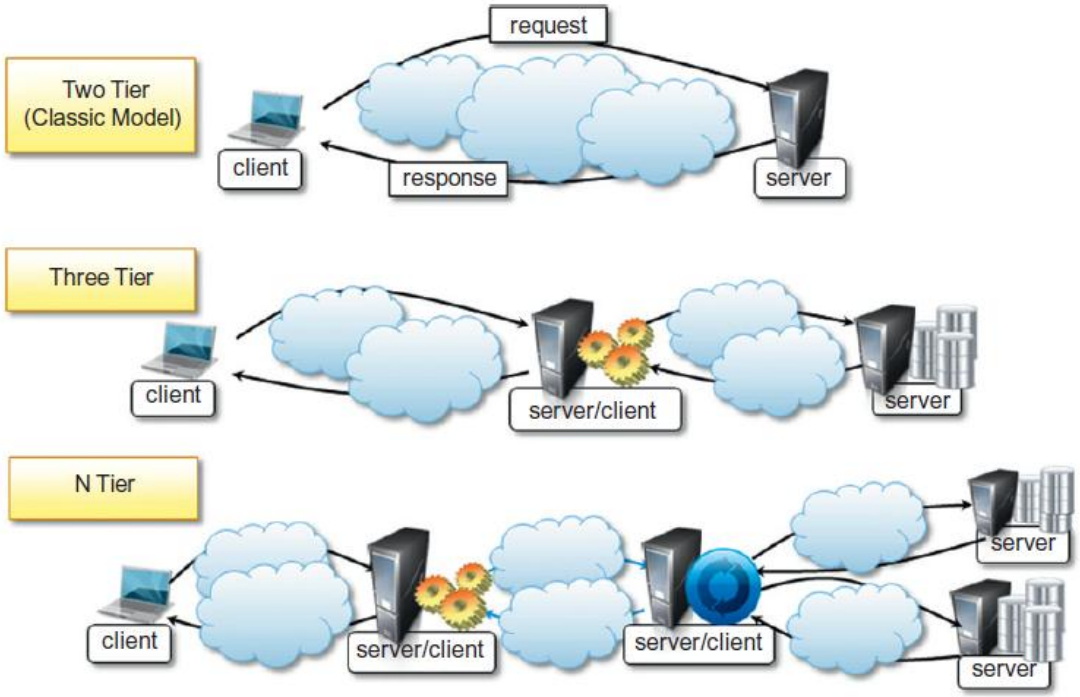
System Architectural Styles

- Deals with physical organization of components and processes over a distributed infrastructure
- Provide a set of reference models for the deployment of distributed systems
- Fundamental reference styles
    - Client/server
    - Peer-to-Peer.

System Architectural Styles

Client/Server

- Client/Server model features a **server** and a **client.**
- Interaction is through a network connection using a given protocol



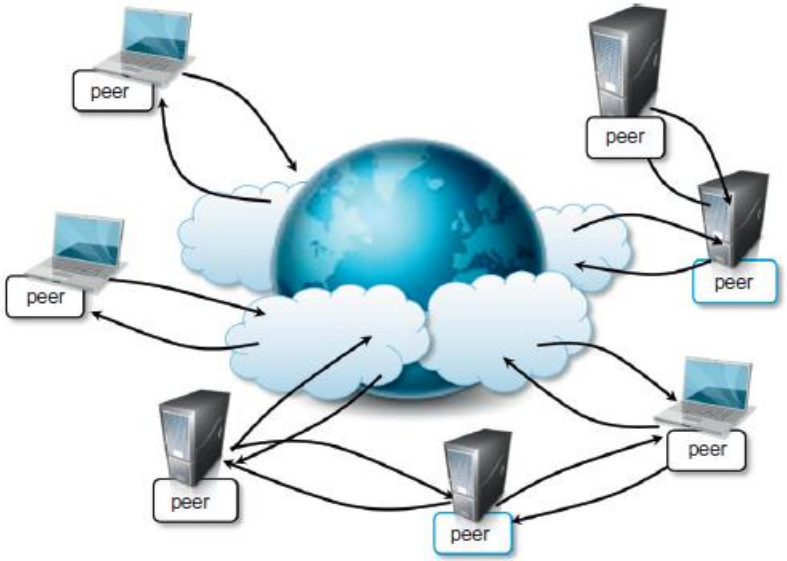
Major Models

- Thin-client model
- Fat-client model

System Architectural Styles

Peer-to-Peer (P2P)

- A symmetric architecture which allows H/W and S/W to communicate without the need for a server
- Each peer acts as a server when it processes requests from other peers and as a client when it issues requests to other peers



Inter Process Communication

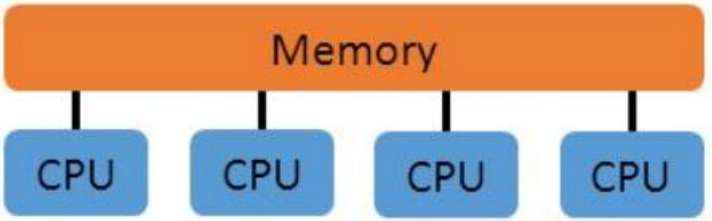
- **Process** is the instance of a computer program that is being executed.
- A process can be classified as
  - Independent process--Not affected by the execution of other processes
  - Co--operating process--Affected by other executing processes
- Inter process communication (**IPC**) is a mechanism which allows **processes** to **communicate** with each other and **synchronize** their **actions**.
- IPC is a method of co--operation between the processes for problem solving.
- Processes can communicate with each other through
  - Shared Memory
  - Message Passing
  - Remote Procedure Call (RPC)

Inter Process Communication

Shared Memory vs. Message Passing

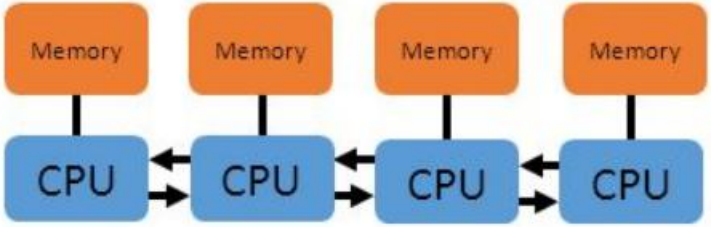
Shared Memory

- Implicit communication via memory operations (load/store/lock)
- Global address space



Message Passing

- Communicate data among a set of processors without the need for a global memory
- Each process has its own local memory and communicate with others using message passing



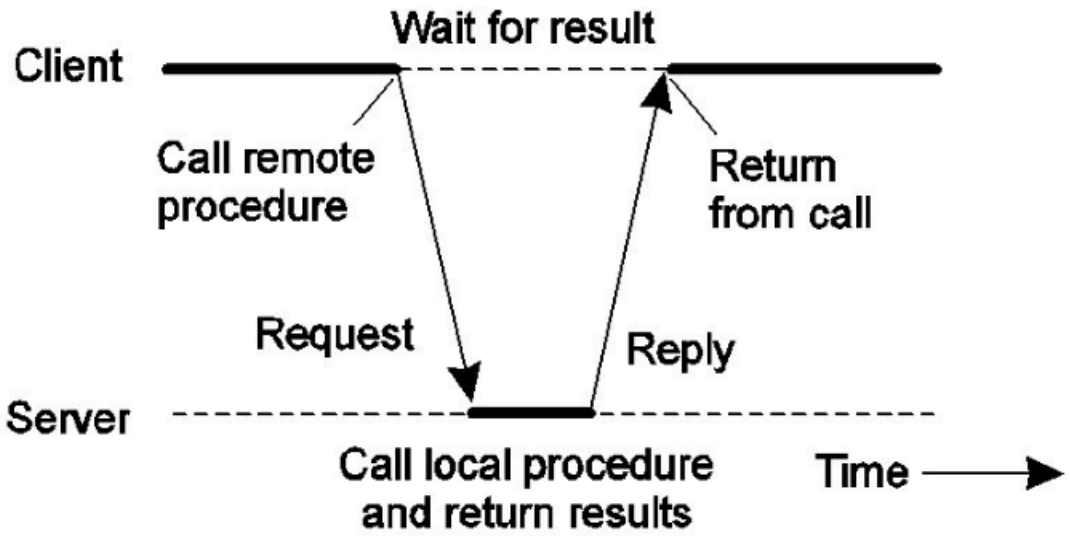
Inter Process Communication

Remote Procedure Call (RPC)

- Protocol for constructing distributed client-server based applications
- An abstraction of inter--process communication
- RPC runtime takes care of the network differences between the processes
- Called procedure need not exist in the same address space as the calling procedure
- Processes may be on the same system, or they may be on different systems with a network connecting them

Inter Process Communication

Basic Remote Procedure Call Operation



Process communication between caller & callee can be hidden using procedure call mechanism

Message based Communication in IPC

- Message-based communication model refers to various model for inter-process communication, which is based on the data streaming abstraction
- Major distributed programming models that uses “message-based communication model”
    - Message Passing
    - Remote Procedure Call
    - Distributed Objects
    - Active Objects
    - Web Services

Message based Communication in IPC

- **Message Passing**
  - The concept of message is the main abstraction of the model
  - Entities exchanging data and information explicitly encode in the form of message
- **Remote Procedure Call**
  - Explores the possibilities of remote procedural call (execution of program in remote processes)
- **Distributed Objects**
  - Implementation of RPC model for the Object-oriented model
  - Contextualizes remote invocation of methods extended by objects
- **Active Objects**
  - The Active object pattern's goal is to separate the method calling from method execution.

Message based Communication in IPC

- **Web Services**
- Provides an implementation of the RPC concept over HTTP
- Allows interaction of components developed with different technologies

Summary:

- Message is a fundamental abstraction of IPC, and is used either explicitly or implicitly
- Principal use of message passing is to define interaction protocols among distributed components for coordinating data exchange activity

Message based Communication Models

- Constitutes a fundamental block for several distributed programming paradigms
- Important to understand how messages are exchanged and among how many distributed components
- Client server model has been identified as reference model for interaction in most cases
- Variations of the client server model allow for different interaction patterns
    - Point-to-point message model
    - Publish-and-subscribe message mode
    - Request-reply message model

Technologies for Distributed Computing

- Relevant technologies that provide concrete implementations of interaction models which mostly rely on message-based communication
    - Remote Procedure Call
    - Distributed Object Framework
- Examples of Distributed Object Framework
    - Common Object Request Broker Architecture (CORBA)
    - Distributed component object model(DCOM/COM1)
    - Java Remote Method Invocation (RMI)
    - .NET remoting

Service Oriented Computing

- Organizes distributed systems in terms of services representing major abstraction for building systems
- Service orientation expresses applications and software systems as aggregations of services that are coordinated within a service-oriented architecture(SOA)
- Web services are the de facto approach for developing SOA
- Web services enables cloud computing systems leverage the Internet as the main interaction channel between users and the system

Service Oriented Computing

What is a service?

- A service encapsulates a software component that provides a set of coherent and related functionalities that can be reused and integrated into bigger and more complex applications
- Major characteristics that identify a service
  - **Boundaries are explicit**--Application composed of services that are spread across different domains, trust authorities, and execution environments
  - **Services are autonomous**--Not designed to be part of a specific system, but can be integrated in several S/W systems at the same time
  - **Services share schema and contracts, not class or interface definitions**
    - Not expressed in terms of classes or interfaces
    - Define themselves in terms of schemas and contracts
  - **Services compatibility is determined based on policy**--Service orientation separates structural compatibility from semantic compatibility

Service Oriented Architecture (SOA)

- SOA organizes a software system into a collection of interacting services
- Encompasses a set of design principles for facilitating system development
- Provides a means for integrating components into a coherent and decentralized system
- Service provider and service consumer are two major roles within SOA

Service Oriented Architecture (SOA)

- A reference model for architecting several software systems, especially enterprise business applications and other systems
- Guiding principles which characterize SOA platforms
    - Standardized Service Contract
    - Loose Coupling
    - Abstraction
    - Reusability
    - Autonomy
    - Lack of State
    - Discoverability
    - Composability

Web Services

- Prominent technology for implementing SOA systems and applications
- Leverage internet technologies and standards for building distributed systems
- Allow interoperability across different platforms and programming languages
- Characteristics that makes web services the technology of choice for SOA
  - Allow for interoperability across different platforms and programming languages
  - Based on well-known and vendor-independent standards such as HTTP, SOAP, XML, and WSDL
  - Provide an intuitive and simple way to connect heterogeneous software systems
  - Provide features required by enterprise business applications to be used in an industrial environment

Acknowledgement

- I would like to thank Dr. Rajkumar Buyya, Dr. Christian Vecchiola and Dr. S. Thamarai Selvi for the figures and course material provided in Mastering Cloud Computing, McGraw Hill Education (India) Private Limited

Thank You