RV College of Engineering
Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi
Approved by AICTE, New Delhi, Accredited By NAAC, Bengaluru And NBA, New Delhi

Cloud Computing and Architecture

Vision of Cloud Computing

UNIT 1

Prof. Rajesh R.M.
Assistant Professor
Dept. of AIML
RV College, Bengaluru

Go change the world

Defining a Cloud

- cloud computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the internet (“the cloud”) to offer faster innovation, flexible resources, and economies of scale.

Vision of Cloud Computing:

Cloud Computing allows renting infrastructure, runtime environments, and services on a pay-per-use basis

The need for ubiquitous storage and compute power on demand is the most common reason to consider cloud computing. A scalable runtime for applications is an attractive option for application and system developers that do not have infra- structure or cannot afford any further expansion of existing infrastructure.

Defining a Cloud

Cloud Computing

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

we can define three criteria to discriminate whether a service is delivered in the cloud computing style:

- The service is accessible via a Web browser (nonproprietary) or a Web services application programming interface (API).
- Zero capital expenditure is necessary to get started.
- You pay only for what you use as you use it.

Types of Cloud

Public Cloud

Private Cloud

Hybrid Cloud

Public clouds are the most common deployment models in which necessary IT infrastructure (e.g., virtualized data centres) is established by a third-party service provider that makes it available to any consumer on subscription basis. Such clouds are appealing to users because they allow users to quickly leverage compute, storage, and application services. In this environment, users' data and applications are deployed on cloud data centres on the vendor's premises.

Private Cloud

Large organizations that own massive computing infrastructures can still benefit from cloud computing by replicating the cloud IT service delivery model in-house.

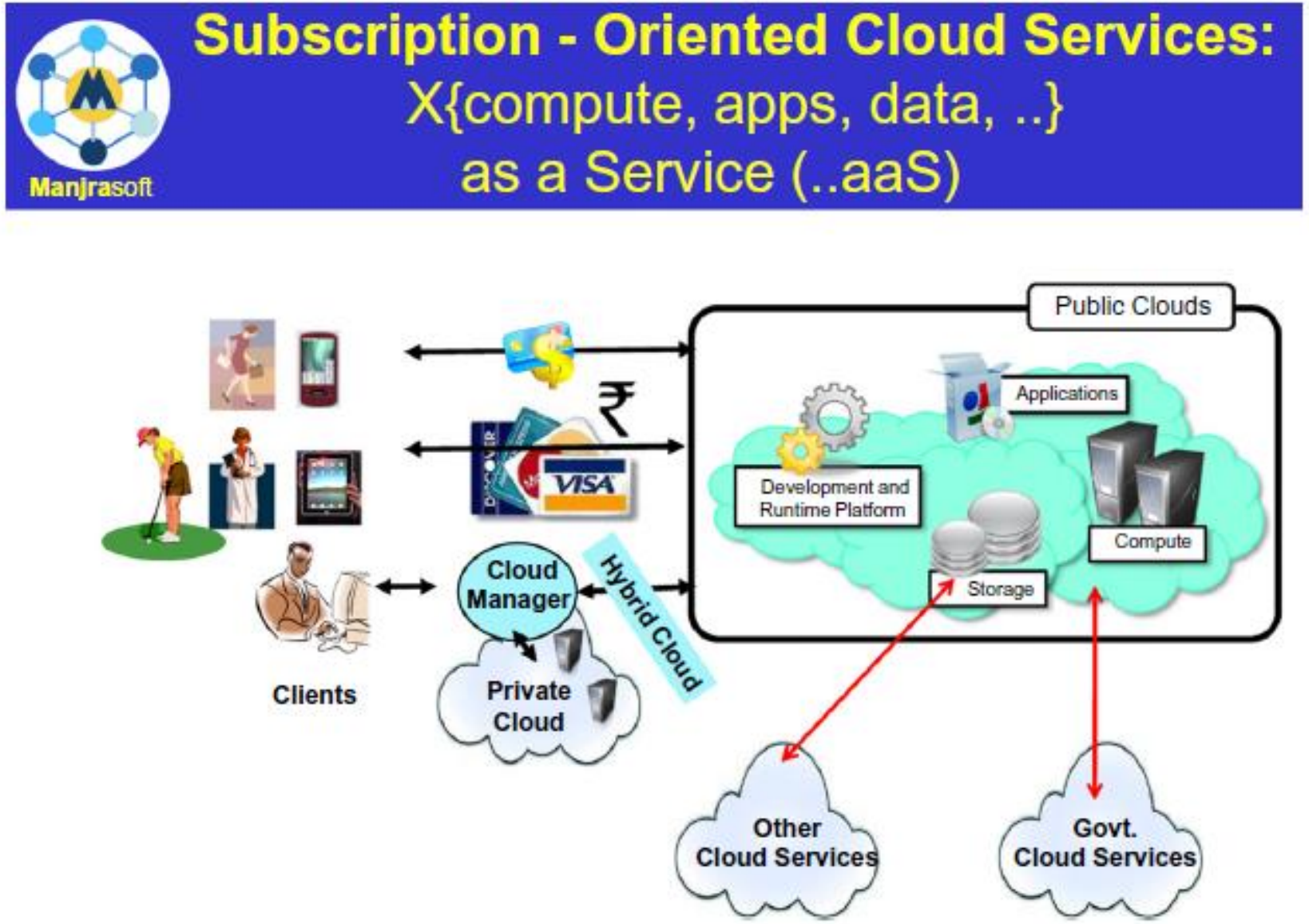
Types of Cloud

Hybrid Cloud

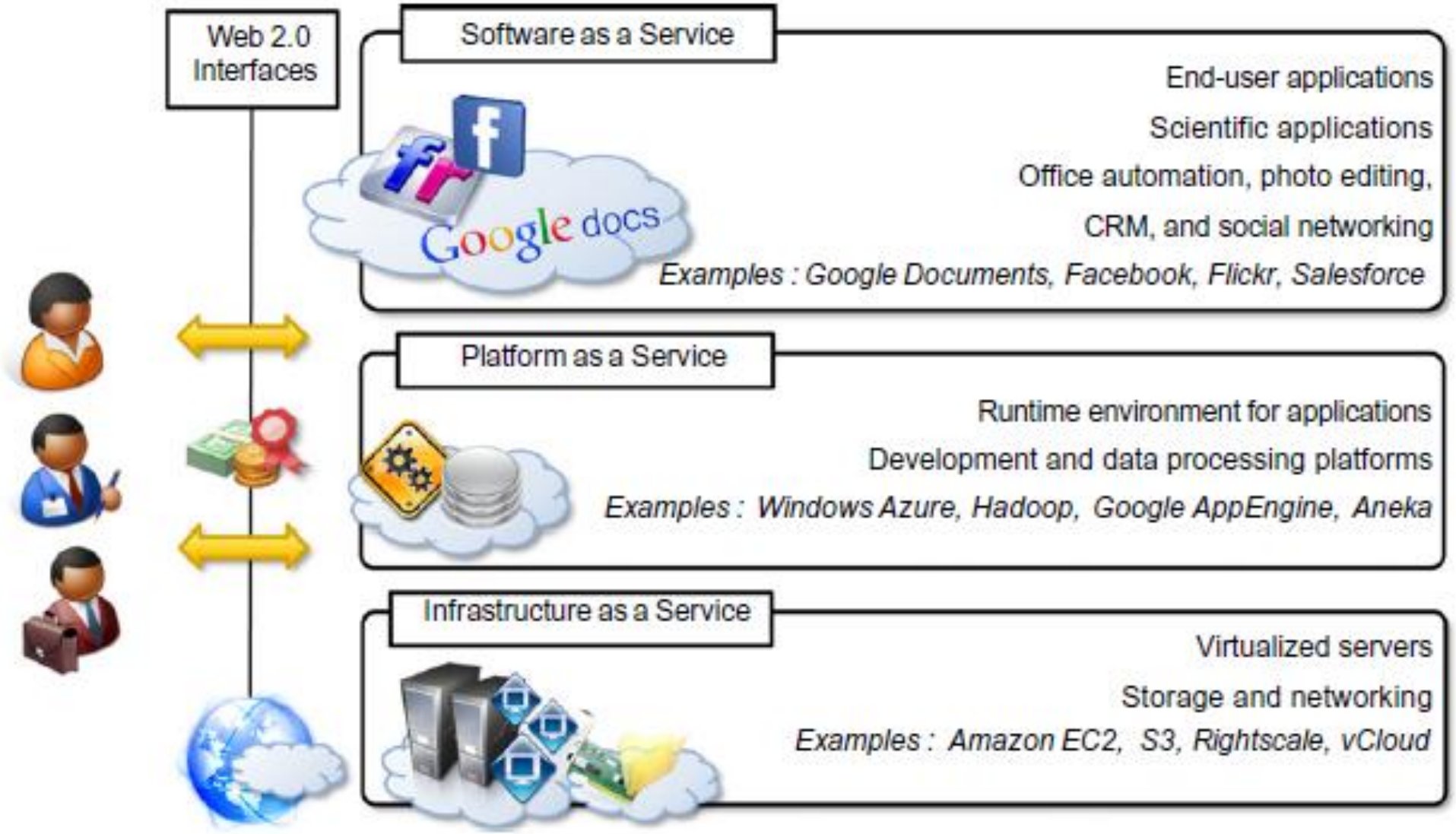
Combination of Public clouds and Private Cloud

Whenever private cloud resources are unable to meet users' quality-of-service requirements, hybrid computing systems, partially composed of public cloud resources and privately owned infra- structures, are created to serve the organization's needs.

Bird Eye View of Cloud Computing



The Cloud Computing Reference Model



The Cloud Computing Reference Model

IAAS (Infrastructure-as-a-Service)

At the base of the stack, Infrastructure-as-a-Service solutions deliver infrastructure on demand in the form of virtual hardware, storage, and networking. Virtual hardware is utilized to provide compute on demand in the form of virtual machine instances. These are created at users' request on the provider's infrastructure, and users are given tools and interfaces to configure the software stack installed in the virtual machine.

PAAS (Platform-as-a-Service)

Platform-as-a-Service solutions are the next step in the stack. They deliver scalable and elastic runtime environments on demand and host the execution of applications. These services are backed by a core middleware platform that is responsible for creating the abstract environment where applications are deployed and executed.

SAAS (Software-as-a-Service)

SAAS (Software-as-a-Service) solutions provide applications and services on demand. Most of the common functionalities of desktop applications—such as office automation, document management, photo editing, and customer relationship management (CRM) software—are replicated on the provider's infrastructure and made more scalable and accessible through a browser on demand.

Characteristics of Cloud

- No up-front commitments
- On-demand access
- Nice pricing
- Simplified application acceleration and scalability
- Efficient resource allocation
- Energy efficiency
- Seamless creation and use of third-party services

Challenges ahead

- Configuration
- Networking
- sizing of cloud computing systems
- Dynamic provisioning of cloud computing services and resources arises
- Virtualization
- Security in terms of confidentiality, secrecy, and protection of data
- Legal issues may also arise

Historical Developments

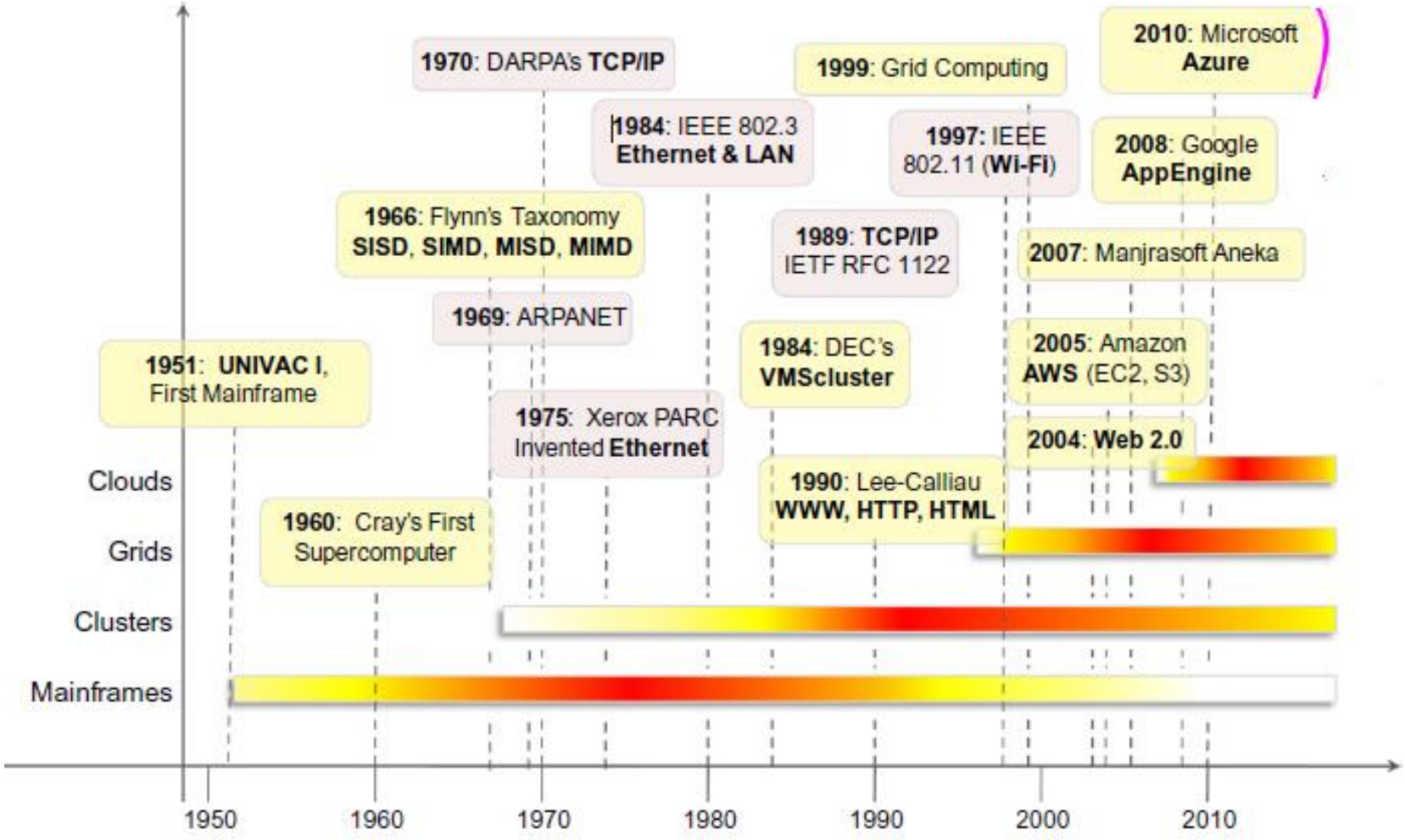
The idea of renting computing services by leveraging large distributed computing facilities has been around for long time. It dates back to the days of the mainframes in the early 1950s. From there on, technology has evolved and been refined. This process has created a series of favourable conditions for the realization of cloud computing.

five core technologies that played an important role in the realization of cloud computing.

- Distributed systems
- Virtualization
- Web 2.0
- service orientation
- utility computing

Distributed Systems

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



Service-oriented computing (SOC)

This approach adopts the concept of services as the main building blocks of application and system development. Service-oriented computing(SOC) supports the development of rapid, low-cost, flexible, interoperable, and evolvable applications and systems.

Service-oriented computing introduces and diffuses two important concepts, which are
Quality of service (QoS)
Software-as-a-Service (SaaS).

Quality of service (QoS)
QoS identifies a set of functional and non-functional attributes that can be used to evaluate the behaviour of a service from different perspectives.

Software-as-a-Service

- SAAS introduces a new delivery model for applications. The term has been inherited from the world of application service providers (ASPs),
- The ASP is responsible for maintaining the infrastructure and making available the application, and the client is freed from maintenance costs and difficult upgrades.
- Web services are software components that expose functionalities accessible using a method invocation pattern that goes over the HyperText Transfer Protocol (HTTP). Web Service Description Language (WSDL) Simple Object Access Protocol (SOAP) [

Utility-oriented computing

- Utility computing is a vision of computing that defines a service-provisioning model for compute services in which resources such as storage, compute power, applications, and infrastructure are packaged and offered on a pay-per-use basis.
- computing grids provided a planet-scale distributed computing infrastructure that was accessible on demand. Computing grids brought the concept of utility computing to a new level: market orientation.
- With utility computing accessible on a wider scale, it is easier to provide a trading infrastructure where grid products—storage, computation, and services—are bid for or sold. Moreover, e-commerce technologies provided the infrastructure support for utility computing.

Building cloud computing environments

1. Application development
2. Infrastructure and system development
3. Computing platforms and technologies

Building cloud computing environments

1. **Application development**

- Applications that leverage cloud computing benefit from its capability to dynamically scale on demand.
- One class of applications that takes the biggest advantage of this feature is that of Web applications. Their performance is mostly influenced by the workload generated by varying user demands

Class of Applications

resource-intensive applications (either data intensive or compute-intensive applications.)

scientific applications (require huge computing capacity to perform large-scale experiments)

Cloud computing provides a solution for on-demand and dynamic scaling across the entire stack of computing. This is achieved by (a) providing methods for renting compute power, storage, and networking; (b) offering runtime environments designed for scalability and dynamic sizing; and (c) providing application services that mimic the behavior of desktop applications but that are completely hosted and managed on the provider side

Building cloud computing environments

Cloud computing provides a solution for on-demand and dynamic scaling across the entire stack of computing.

This is achieved by

(a) providing methods for renting compute power, storage, and networking;

(b) offering runtime environments designed for scalability and dynamic sizing;

(c) providing application services that mimic the behaviour of desktop applications but that are completely hosted and managed on the provider side

Building cloud Computing environments

Infrastructure and system development

Distributed computing is a foundational model for cloud computing because cloud systems are distributed systems.

Web 2.0 technologies constitute the interface through which cloud computing services are delivered, managed, and provisioned.

Cloud computing is often summarized with the acronym XaaS—Everything-as-a-Service—that clearly underlines the central role of service orientation

Virtualization is another element that plays a fundamental role in cloud computing. This technology is a core feature of the infrastructure used by cloud providers

These are all considerations that influence the way we program applications and systems based on cloud computing technologies.

Computing platforms and technologies

Amazon web services (AWS)

Google AppEngine

Microsoft Azure

Hadoop

Force.com and Salesforce.com

Manjrasoft Aneka

Computing platforms and technologies

Amazon web services (AWS)

- AWS offers comprehensive cloud IaaS services ranging from virtual compute, storage, and networking to complete computing stacks.
- AWS is mostly known for its compute and storage on demand services, namely Elastic Compute Cloud (EC2) and Simple Storage Service (S3)
- EC2 provides users with customizable virtual hardware that can be used as the base infrastructure for deploying computing systems on the cloud
- EC2 instances are deployed either by using the AWS console, which is a comprehensive Web portal for accessing AWS services, or by using the Web services API available for several programming languages
- EC2 also provides the capability to save a specific running instance as an image, thus allowing users to create their own templates for deploying systems. These templates are stored into S3 that delivers persistent storage on demand.

Computing platforms and technologies

Google AppEngine

- Google AppEngine is a scalable runtime environment mostly devoted to executing Web applications. These take advantage of the large computing infrastructure of Google to dynamically scale as the demand varies over time
- Developers can build and test applications on their own machines using the AppEngine software development kit (SDK), which replicates the production runtime environment and helps test and profile applications.
- s. Once development is complete, developers can easily migrate their application to AppEngine, set quotas to contain the costs generated, and make the application available to the world.
- The languages currently supported are Python, Java, and Go.

Computing platforms and technologies

Microsoft Azure

- Microsoft Azure is a cloud operating system and a platform for developing applications in the cloud.
- It provides a scalable runtime environment for Web applications and distributed applications in general.
- Applications in Azure are organized around the concept of roles, currently, there are three types of role: **Web role, worker role, and virtual machine role.**
- The Web role is designed to host a Web application
- The worker role is a more generic container of applications and can be used to perform workload processing
- virtual machine role provides a virtual environment in which the computing stack can be fully customized, including the operating systems.

Computing platforms and technologies

Hadoop

- Apache Hadoop is an open-source framework that is suited for processing large data sets on commodity hardware.
- Hadoop is an implementation of MapReduce, an application programming model developed by Google, which provides two fundamental operations for data processing: map and reduce.
- The former transforms and synthesizes the input data provided by the user; the latter aggregates the output obtained by the map operations.

Computing platforms and technologies

Force.com and Salesforce.com

- Force.com is a cloud computing platform for developing social enterprise applications.
- The platform is the basis for SalesForce.com, a Software-as-a-Service solution for customer relationship management.
- Force.com allows developers to create applications by composing ready-to-use blocks; a complete set of components supporting all the activities of an enterprise are available.
- It is also possible to develop your own components or integrate those available in AppExchange into your applications.

Computing platforms and technologies

Manjrasoft Aneka

- Manjrasoft Aneka is a cloud application platform for rapid creation of scalable applications and their deployment on various types of clouds in a seamless and elastic manner.
- It supports a collection of programming abstractions for developing applications and a distributed runtime environment that can be deployed on heterogeneous hardware (clusters, networked desktop computers, and cloud resources).
- Developers can choose different abstractions to design their application: tasks, distributed threads, and map-reduce. These applications are then executed on the distributed service-oriented runtime environment, which can dynamically integrate additional resource on demand

Eras of Computing

Eras of computing

The two fundamental and dominant models of computing are

1. **Sequential Computing**
2. **Parallel Computing**

- The sequential computing era began in the 1940s.
- The interest in parallel computing dates back to the late 1950's, with advancements surfacing in the form of supercomputers throughout the 60's and 70's.

The four key elements of computing developed during these eras are

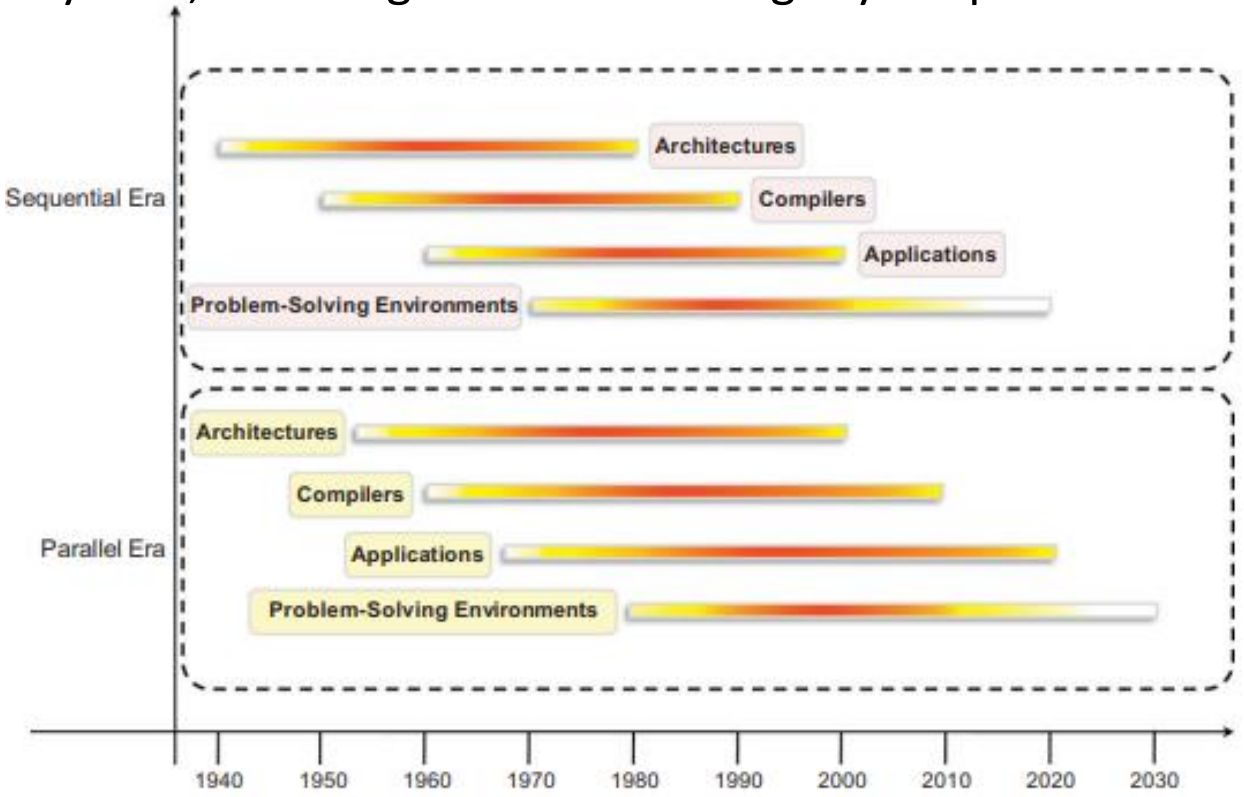
1. **Architectures**
2. **Compilers**
3. **Applications**
4. **Problem Solving Environments**

Parallel vs. distributed computing

Parallel vs. distributed computing

The terms parallel computing and distributed computing are often used interchangeably, even though they mean slightly different things.

- The term parallel implies a tightly coupled system.
- Distributed refers to a wider class of system, including those that are tightly coupled.



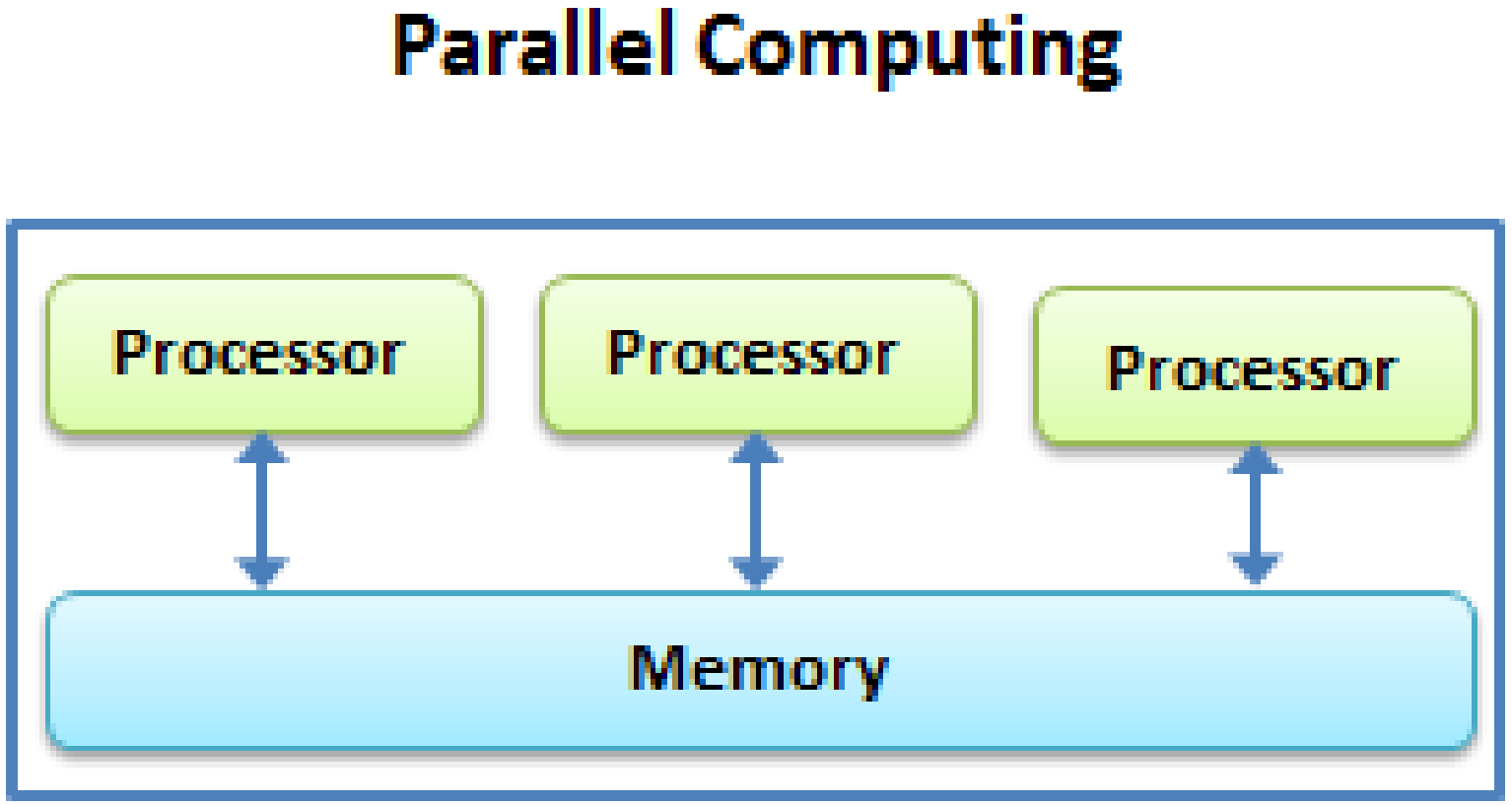
Parallel vs. distributed computing

Parallel computing

The term **parallel computing** refers to a model in which the **computation is divided among several processors sharing the same memory**.

- 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 the shared memory.
- 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.

Parallel vs. distributed computing



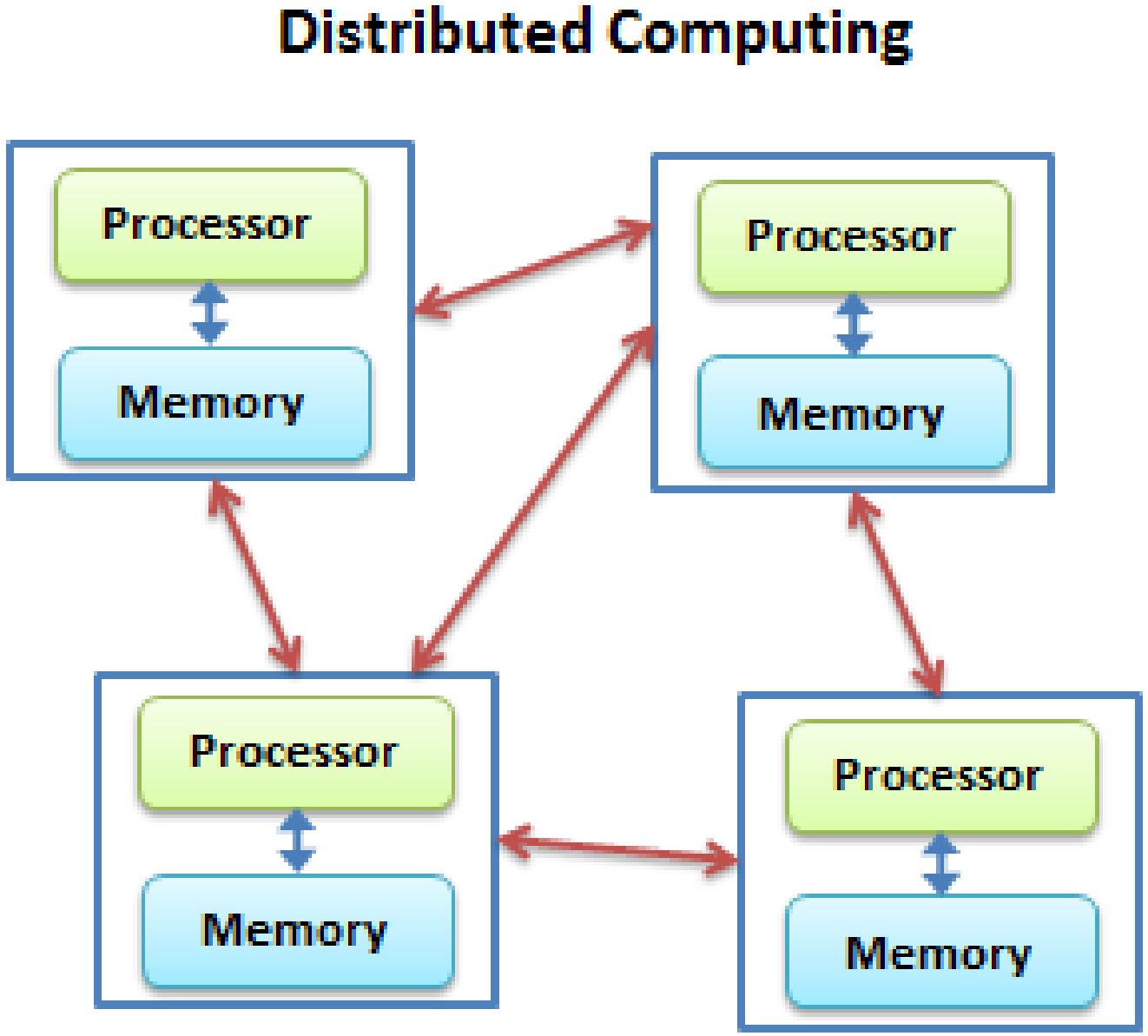
Parallel vs. distributed computing

Distributed computing

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** or **processors on the same computer** or **cores within the same processor**
- 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**
- Distributed computing includes a **wider range of systems** and **applications** than parallel computing and is often considered a more general term.

Parallel vs. distributed computing



Parallel vs. distributed computing

Elements of parallel computing

Processing of multiple tasks simultaneously on multiple processors is called parallel processing.

- 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).
- 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.

Parallel vs. distributed computing

Elements of parallel computing

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.
- Sequential architectures are reaching **physical limitations** as they are constrained by the speed of light and thermodynamics laws.
- Hardware improvements in **pipelining**, **superscalar,** and the **like are non scalable** and require sophisticated **compiler technology.**
- Vector processing works well for certain kinds of problems. (Scientific problems)
- Parallel processing is **mature and can be exploited commercially**
- Significant development in networking technology is paving the way for **heterogeneous computing.**

Parallel vs. distributed computing

Hardware architectures for parallel processing

The core elements of parallel processing are CPUs. Based on the number of instruction and data streams that can be processed simultaneously, computing systems are classified into the following four categories:

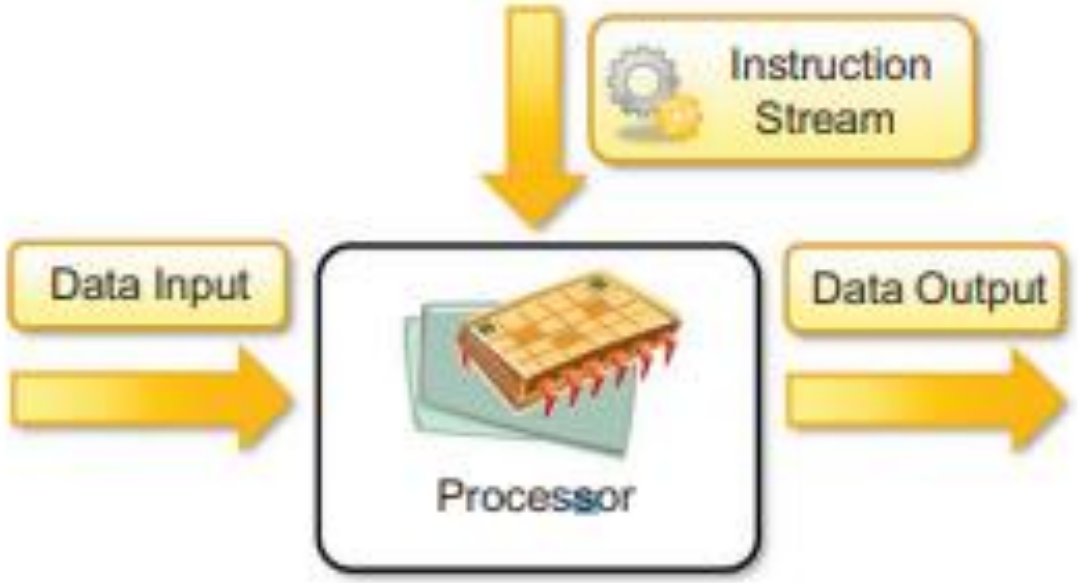
- Single-instruction, single-data (SISD) systems
- Single-instruction, multiple-data (SIMD) systems
- Multiple-instruction, single-data (MISD) systems
- Multiple-instruction, multiple-data (MIMD) systems

Parallel vs. distributed computing

Single-instruction, single-data (SISD) systems

An SISD computing system is a **uniprocessor machine** capable of executing a **single instruction**, which operates on a single data stream

- In SISD, machine instructions are processed **sequentially**. (sequential computers)
- All the instructions and data to be processed have to be **stored in primary memory.**
- The speed of the processing element in the SISD model is **limited.**



Parallel vs. distributed computing

Single-instruction, multiple-data (SIMD) systems

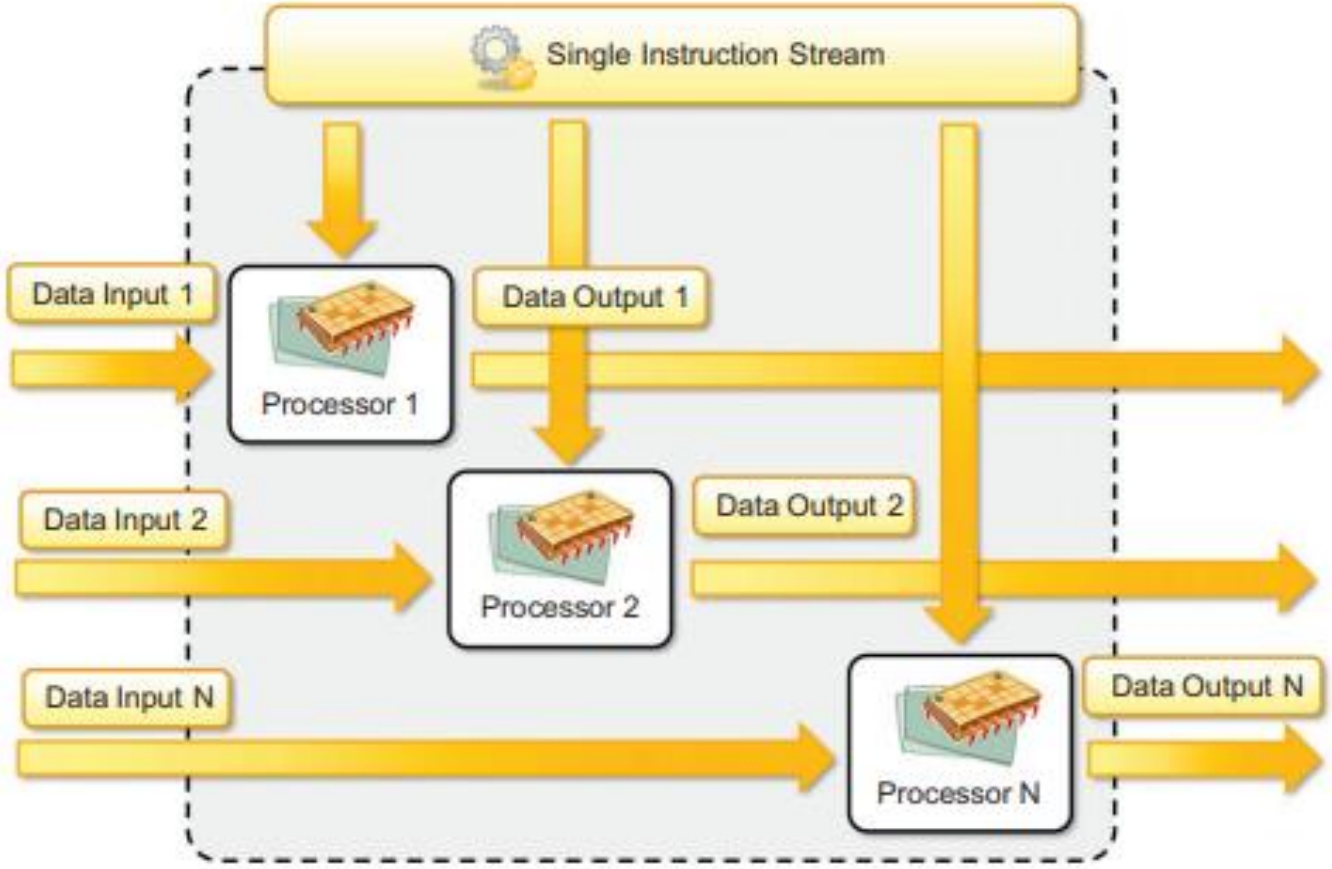
An SIMD computing system is **a multiprocessor** machine capable of executing the **same instruction on all the CPUs** but operating on **different data streams.**

- SIMD model are well suited to scientific computing since they involve lots of vector and matrix operations.

Example

`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);

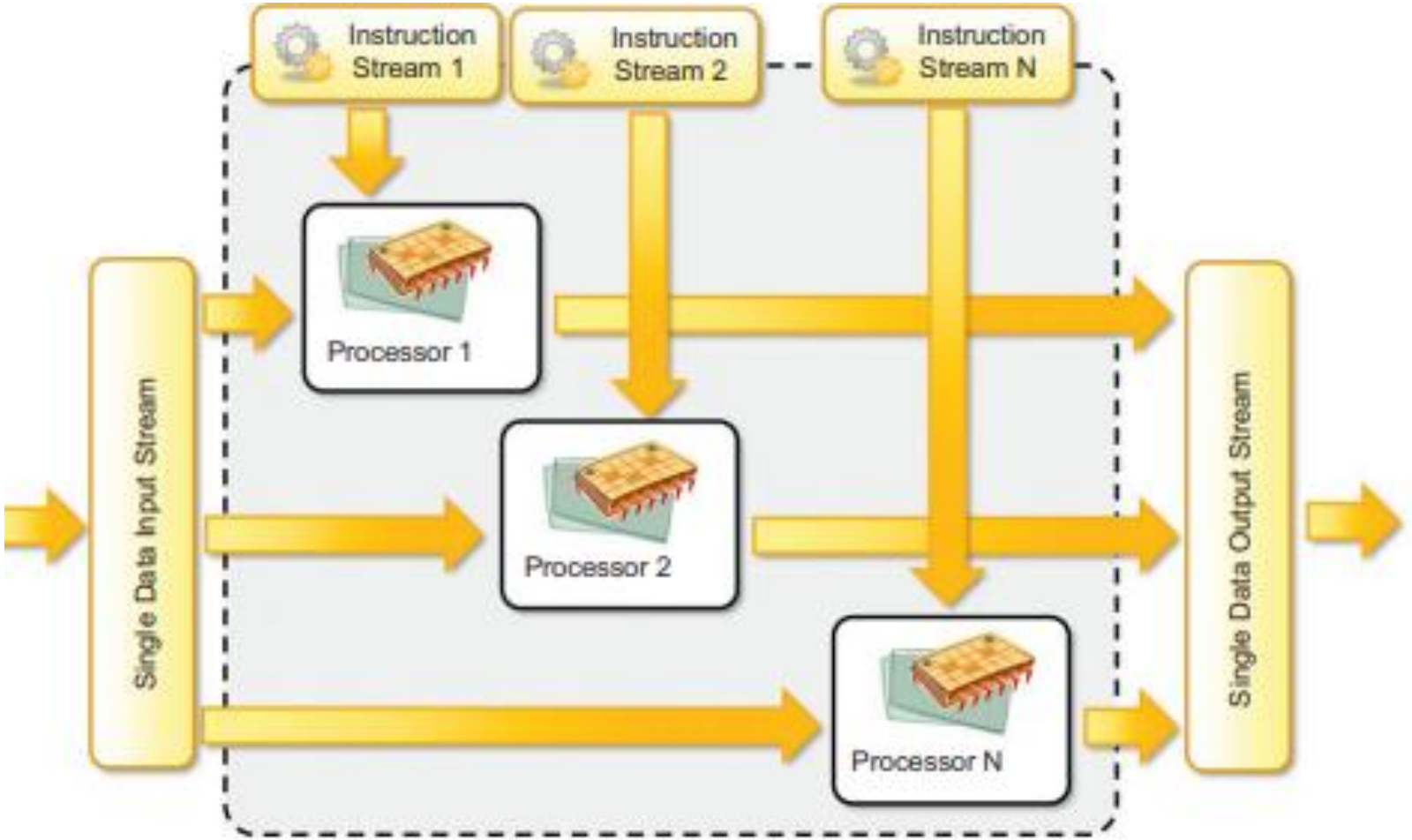


Parallel vs. distributed computing

Multiple-instruction, single-data (MISD) systems

An MISD computing system is a multiprocessor machine capable of executing different instructions on different PEs but all of them operating on the same data set

$$y = \sin(x) + \cos(x) + \tan(x)$$

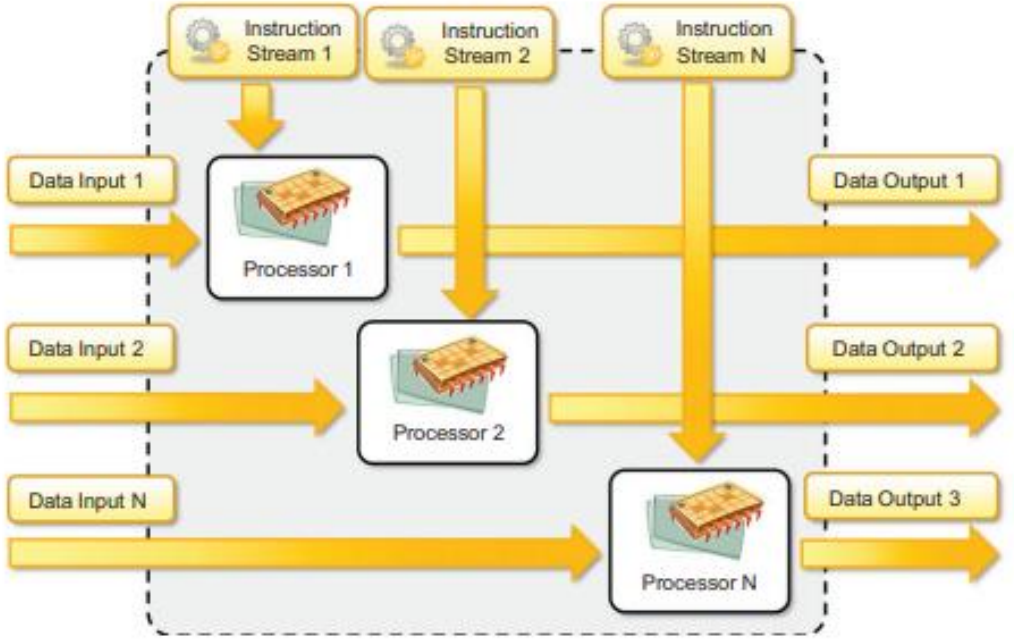


Parallel vs. distributed computing

Multiple-instruction, multiple-data (MIMD) systems

An MIMD computing system is a multiprocessor machine capable of executing multiple instructions on multiple data sets

- Each PE in the MIMD model has separate instruction and data streams.
- MIMD machines are broadly categorized into shared-memory MIMD and distributed-memory MIMD based on the way PEs are coupled to the main memory

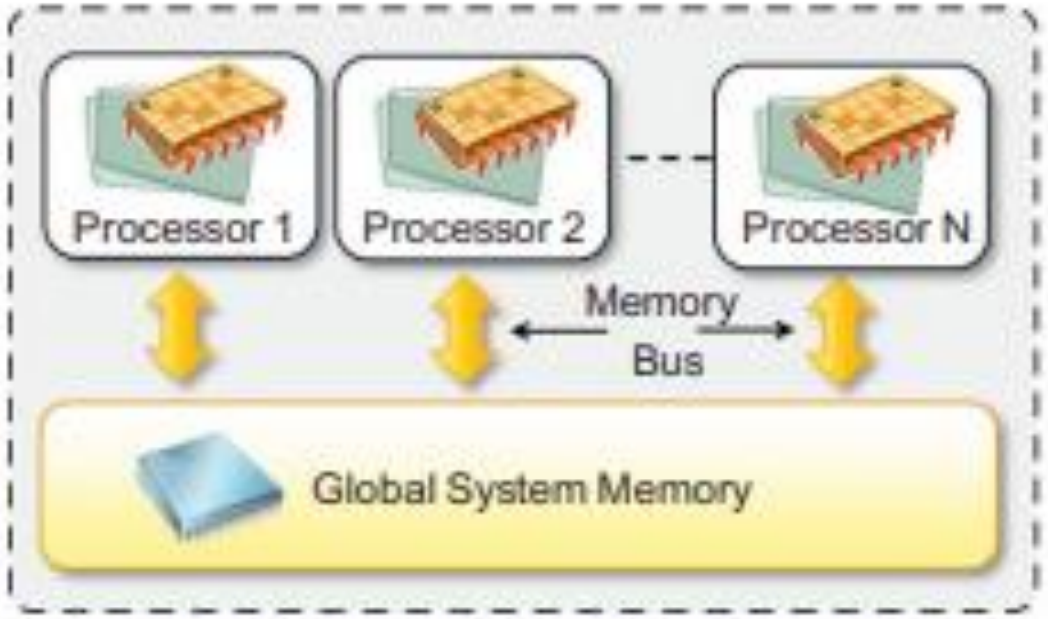


Parallel vs. distributed computing

Multiple-instruction, multiple-data (MIMD) systems

Shared memory MIMD machines

- In the shared memory MIMD model, 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 multiprocessor 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.
- MIMD systems are Silicon Graphics machines and Sun/IBM's SMP (Symmetric Multi-Processing).



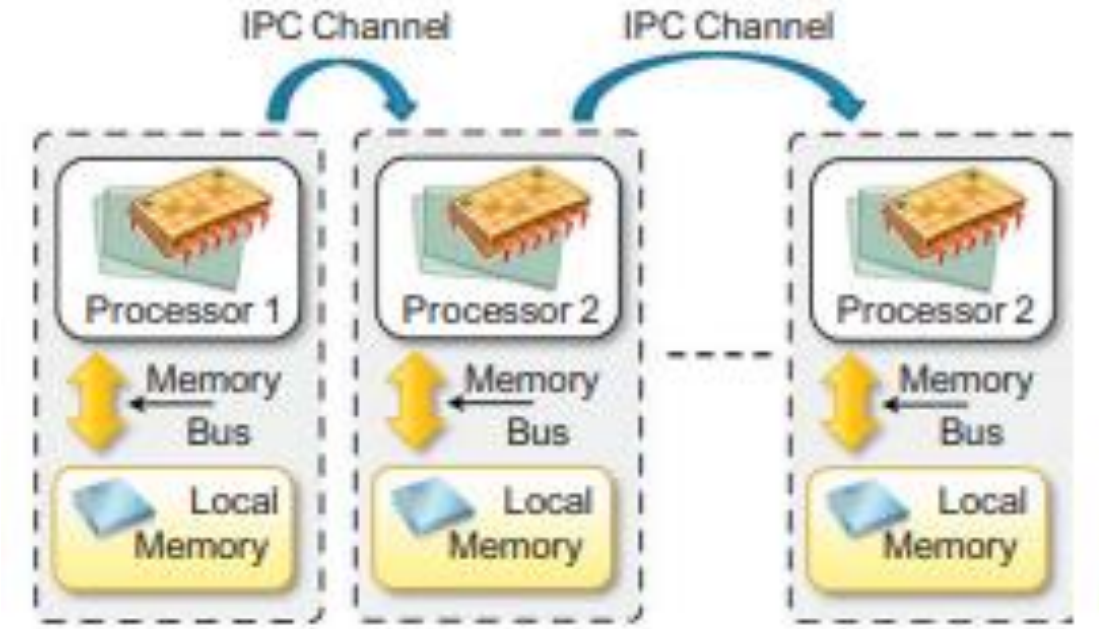
Parallel vs. distributed computing

Multiple-instruction, multiple-data (MIMD) systems

Distributed memory MIMD machines

In the distributed memory MIMD model, all PEs have a local memory. Systems based on this model are also called loosely coupled multiprocessor systems.

- The communication between PEs in this model takes place through the interconnection network
- The network connecting PEs can be configured to tree, mesh, cube
- Each PE operates asynchronously, and if communication/synchronization among tasks is necessary, they can do so by exchanging messages between them.



Approaches to Parallel Programming

Parallel programming approaches

- **Data parallelism -** In the case of data 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.
- **Process parallelism -** In the case of process parallelism, a **given operation has multiple** (but distinct) activities that can be processed **on multiple processors.**
- **Farmer-and-worker model -** In the case of the 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**

Approaches to Parallel Programming

Levels of parallelism

Large grain (or task level)

Medium grain (or control level)

Fine grain (data level)

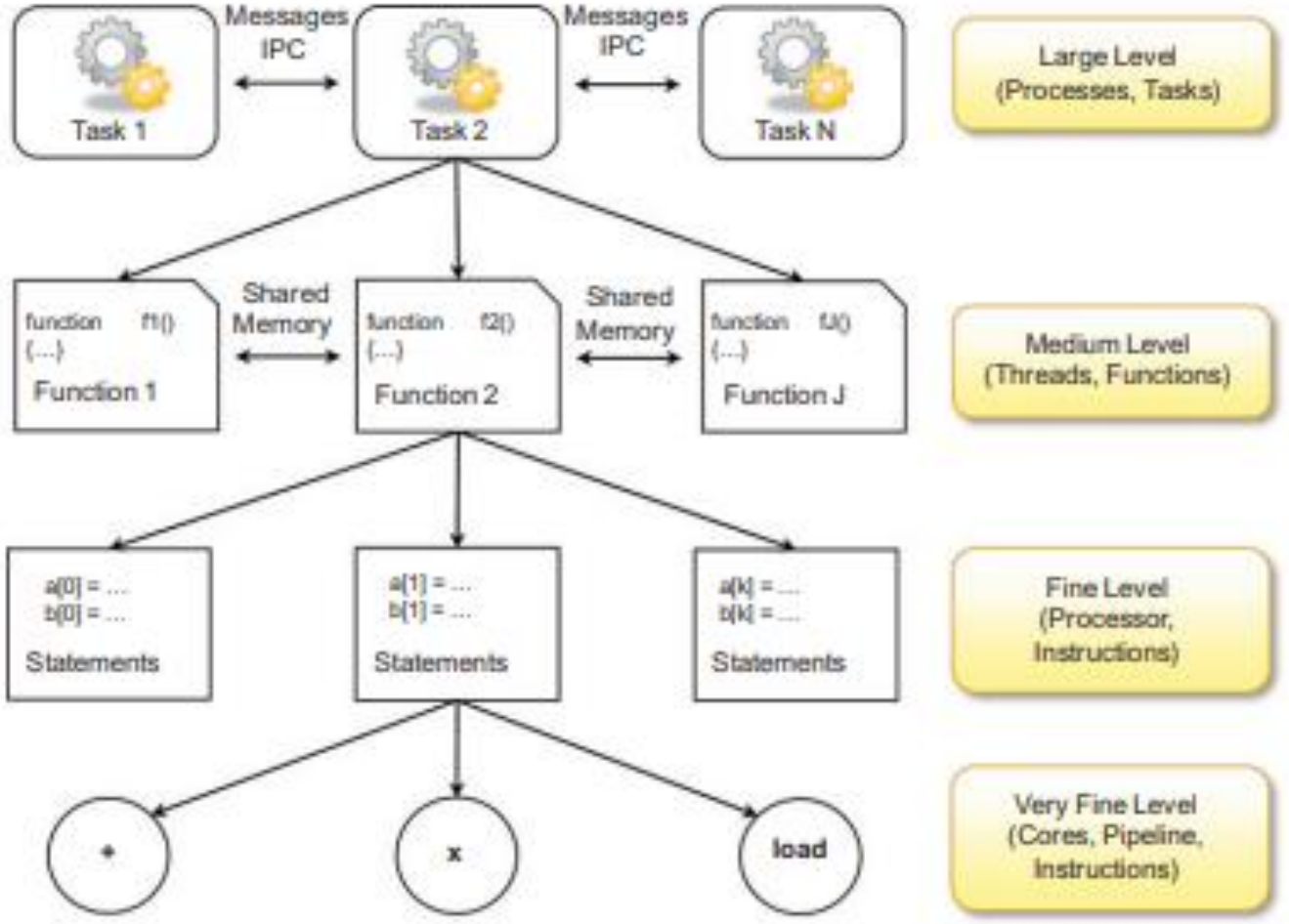
Very fine grain (multiple-instruction issue)

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

Approaches to Parallel Programming

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



Elements of distributed computing

Elements of distributed computing

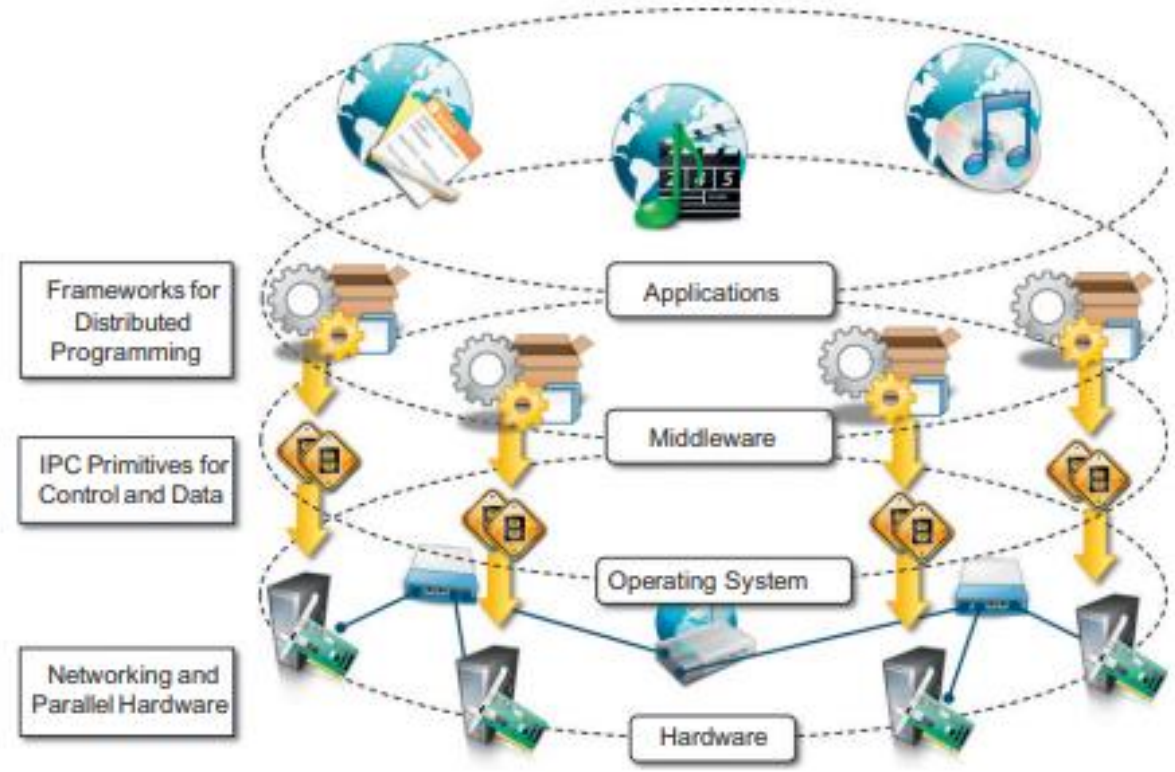
A distributed system is a collection of independent computers that appears to its users as a single coherent system.

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

Elements of distributed computing

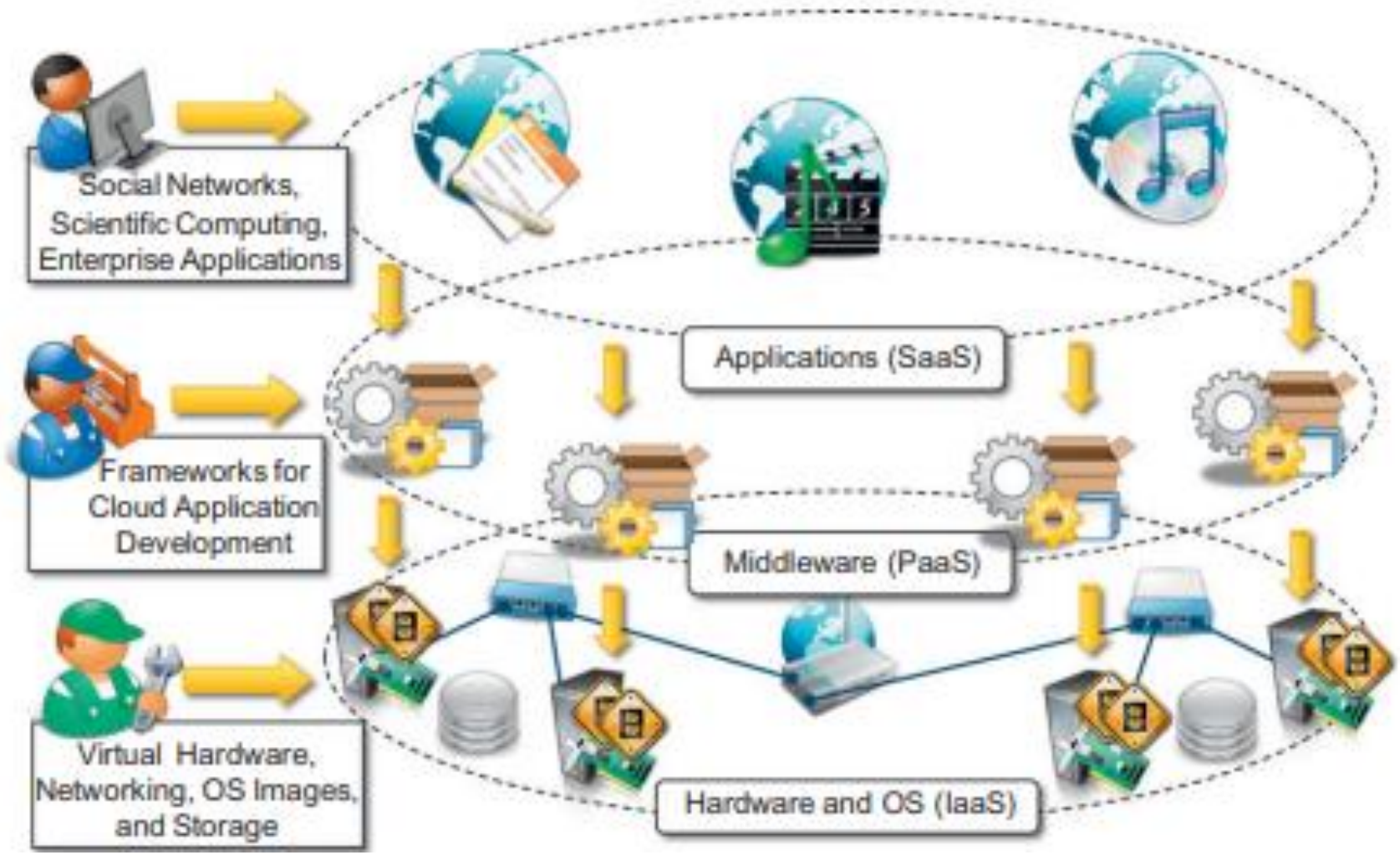
Components of a distributed system

- A distributed system is the result of the interaction of several components that traverse the entire computing stack from hardware to software.
- It emerges from the collaboration of several elements that—by working together—give users the illusion of a single coherent system.



Elements of distributed computing

Cloud Computing distributed system



Elements of distributed computing

Architectural styles for distributed computing

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

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.

Design patterns help in creating a common knowledge within the community of software engineers and developers as to how to structure the relations of components within an application and understand the internal organization of software applications.

Elements of distributed computing

Architectural styles for distributed computing

two major classes architectural styles :

- Software architectural styles
- System architectural styles

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.

Table 2.2 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

Elements of distributed computing

Architectural styles for distributed computing

Data cantered architectures

These architectures identify the data as the fundamental element of the software system, and access to shared data is the core characteristic of the data-centered architectures.

Most Common Architectural Styles

Repository - It is characterized by two main components: the central data structure, which represents the current state of the system, and a collection of independent components, which operate on the central data.

Blackboard - The blackboard architectural style is characterized by three main components:

Knowledge sources. These are the entities that update the knowledge base that is maintained in the blackboard.

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.

Elements of distributed computing

Architectural styles for distributed computing

Data-flow architectures

In the case of data-flow architectures, it is the availability of data that controls the computation. With respect to the data-centered styles, in which the access to data is the core feature, data-flow styles explicitly incorporate the pattern of data flow

Batch Sequential Style. The batch sequential style is characterized by an ordered sequence of separate programs executing one after the other.

Pipe-and-Filter Style. The pipe-and-filter style 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

Elements of distributed computing

Architectural styles for distributed computing

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

Rule-Based Style. This architecture is characterized by representing the abstract execution environment as an inference engine

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

Interpreter Style. The core feature of the interpreter style is the presence of an engine that is used to interpret a pseudo-program expressed in a format acceptable for the interpreter

This model is quite useful in designing virtual machines for high-level programming (Java, C#) and scripting languages (Awk, PERL, and so on).

Elements of distributed computing

Architectural styles for distributed computing

Call & return architectures

This category identifies all systems that are organised 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 of one or more operations.

Top-Down Style. This architectural style is quite representative of systems developed with imperative programming, which leads to a divide-and-conquer approach to problem resolution.

Object-Oriented Style. This architectural style encompasses a wide range of systems that have been designed and implemented by leveraging the abstractions of object-oriented programming (OOP).

Layered Style. The layered system style allows the design and implementation of software systems in terms of layers, which provide a different level of abstraction of the system. Each layer generally operates with at most two layers: the one that provides a lower abstraction level and the one that provides a higher abstraction layer.

Elements of distributed computing

Architectural styles for distributed computing

Architectural styles based on independent components

This class of architectural style models systems in terms of independent components that have their own life cycles, which interact with each other to perform their activities.

There are two major categories

Communicating Processes. In this architectural style, components are represented by independent processes that leverage IPC facilities for coordination management. This is an abstraction that is quite suitable to modelling distributed systems that, being distributed over a network of computing nodes, are necessarily composed of several concurrent processes.

Event Systems. In this architectural style, the components of the system are loosely coupled and connected. In addition to exposing operations for data and state manipulation, each component also publishes (or announces) a collection of events with which other components can register.

Event-based systems have become quite popular, and support for their implementation is provided either at the **API level or the programming language level**.

Elements of distributed computing

System architectural styles

System architectural styles cover the physical organization of components and processes over a distributed infrastructure.

two fundamental reference styles: client/server and peer-to-peer.

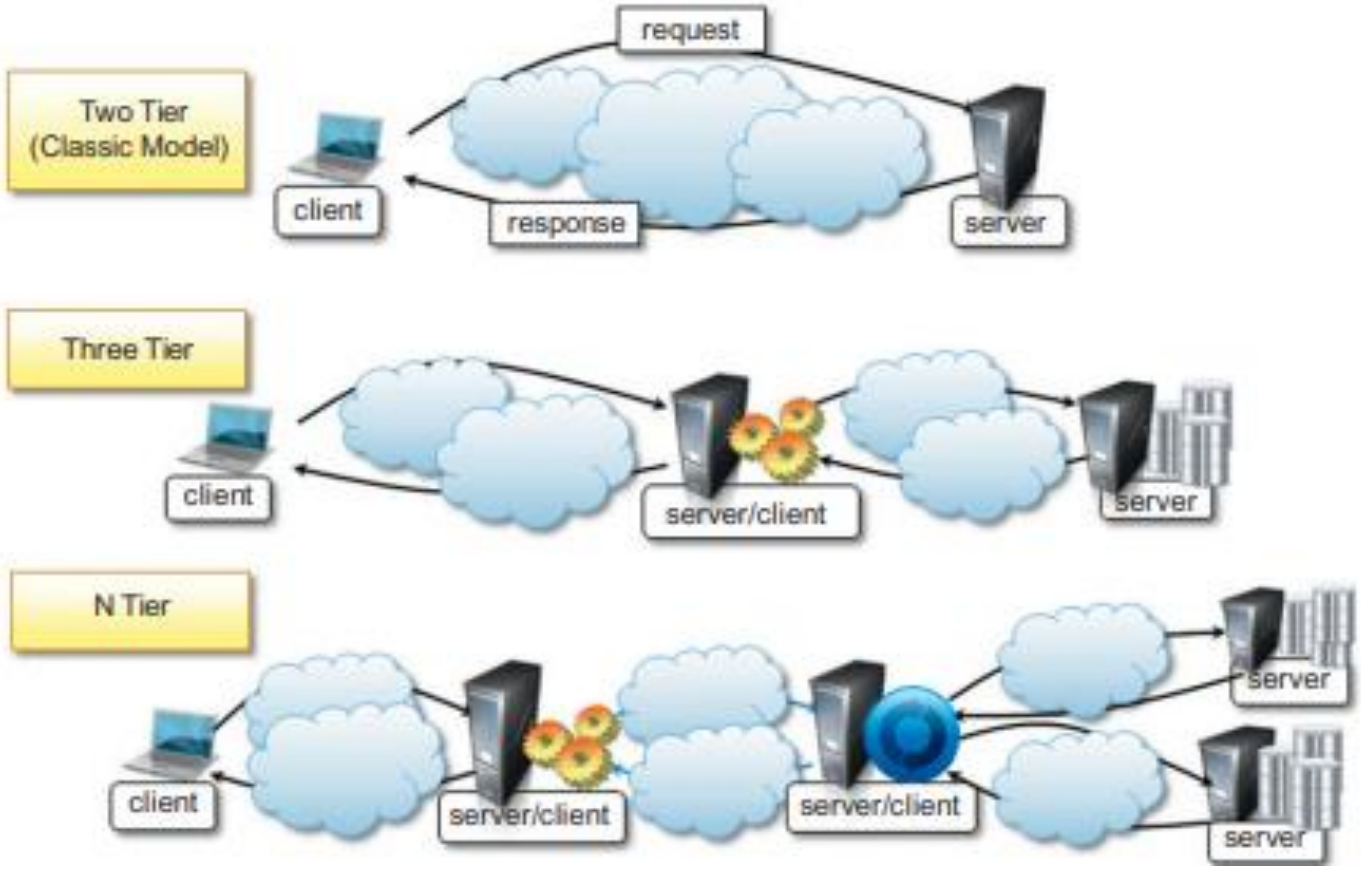
Elements of distributed computing

System architectural styles

Client/server

- This architecture is very popular in distributed computing and is suitable for a wide variety of applications.

Fig. Client/server architectural styles.



- The client/server model features two major components: a server and a client. These two components interact with each other through a network connection using a given protocol.

Elements of distributed computing

System architectural styles

The peer-to-peer model

The peer-to-peer mode architecture 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

The most relevant example of peer-to-peer systems is constituted by file-sharing applications such as Gnutella, BitTorrent

Thank You

Go, Change the world