CSE 423
Cloud Computing and Virtualisation

Parallel and Distributed Computing

Slides 3

Instructor: Asst. Prof. Dr. Hüseyin ABACI

Contents

1. Concurrency, coordination, and communication.
2. Diner philosophers problem.
3. Concurrency and communication.
4. Parallel vs. distributed computing.
5. Hardware architectures for parallel processing.
6. Middleware.
7. Remote procedure call.
8. Distributed object frameworks.
9. Service Oriented Architecture.
10. Web-services.

Dan Marinescu, Cloud Computing: Theory and Practice, 2nd Edition, Morgan Kaufmann, 2017.

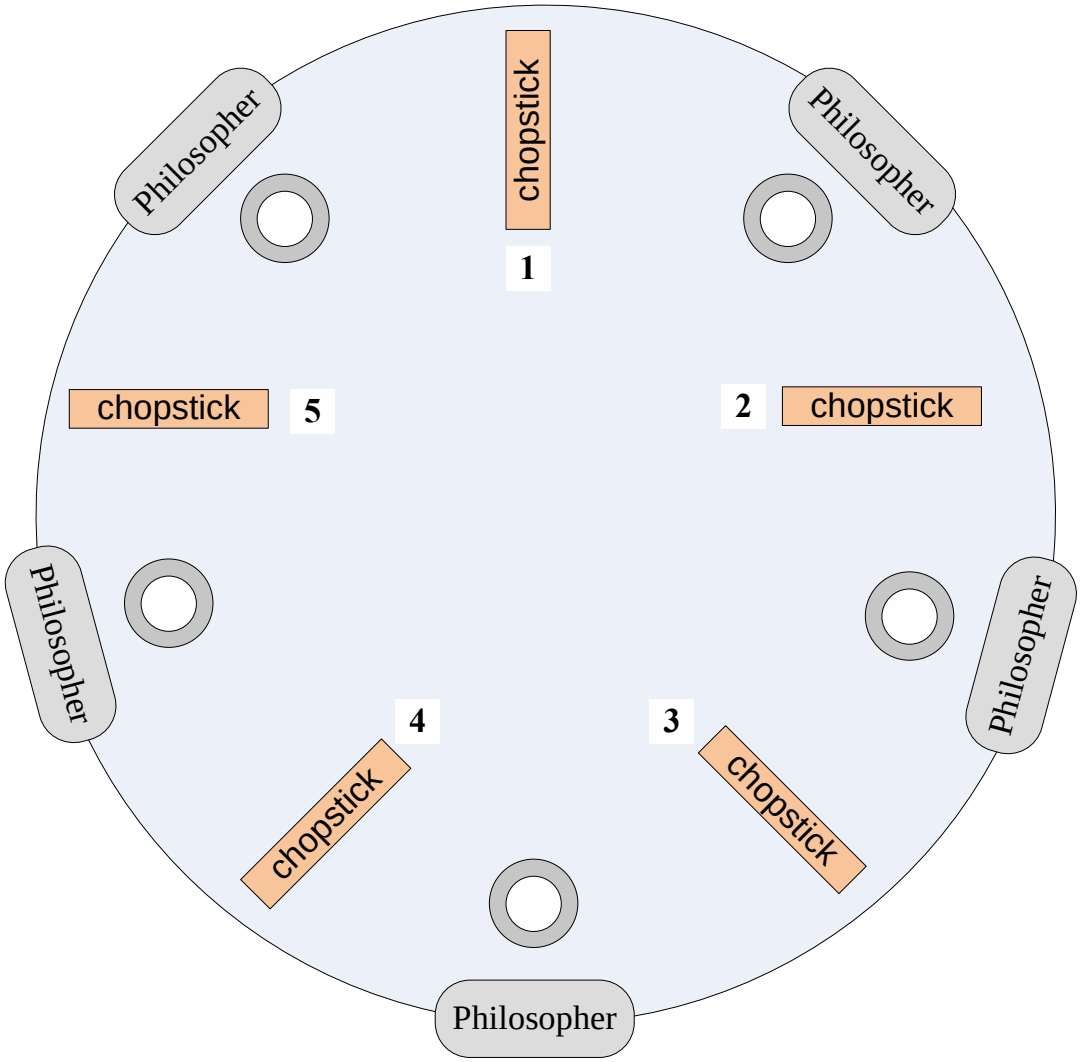
Rajkumar Buyya, Christian Vecchiola, S.Thamarai Selvi , Mastering Cloud Computing: Foundations and Applications Programming, 2013.

Concurrency, coordination, and communication

- Execution of multiple activities in parallel requires some form of communication is necessary for coordination of concurrent activities.
- Coordination is ubiquitous in our daily life; the lack of coordination has negative implications, e.g., deadlocks!
- Communication
  - Affects the overall efficiency of concurrent activities and could significantly increase the completion time of a complex task and even hinder the completion of the task.
  - Requires prior agreement on the communication discipline described by a communication protocol.

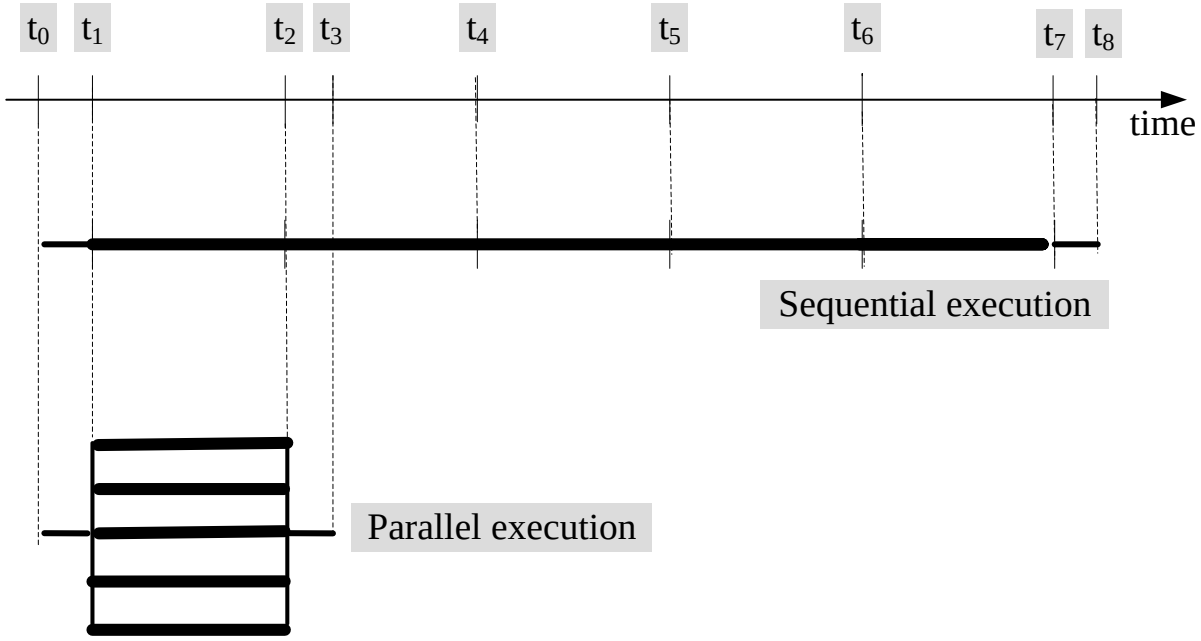
Synchronization of concurrent activities

- Synchronization is a defining aspect of concurrency.
- Diner philosophers problem:
  - Five philosophers sitting at a table alternately think and eat.
  - A philosopher needs the two chopsticks placed left and right of her plate to eat.
  - After finishing eating a philosopher must place the chopsticks back on the table to give a chance to left and right neighbors to eat.
- The problem is nontrivial, the naive solution when each philosopher picks up the chopstick to the left, and waits for the one to the right to become available, or vice versa, fails because it allows the system to reach a deadlock state, in which no progress is possible.



Solution should contain Locking mechanisim to avoid Deadlock .

Speedup



Sequential versus parallel execution of an application. The sequential execution starts at time t_0 goes through a brief initialization phase till time t_1, then starts the actual image processing. When all images have been processes it enters a brief termination phase at time t_7, and finishes at time t_8.

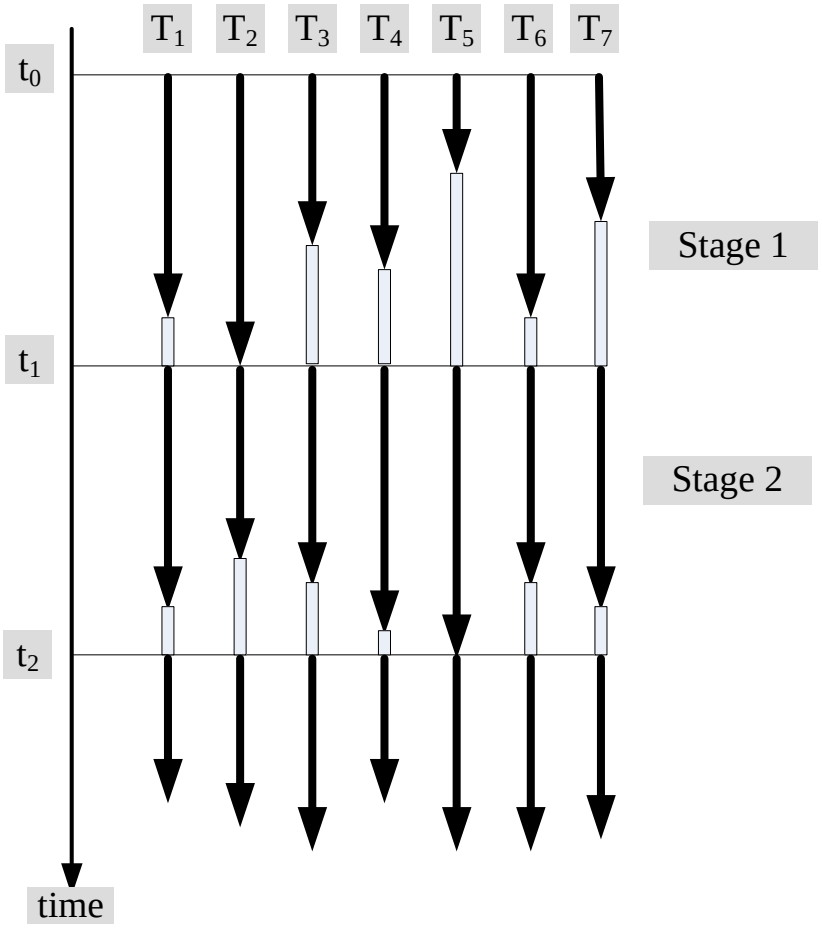
The concurrent/parallel execution has its own brief initialization and termination phases, the actual image processing starts at time t_1 and ends at time t_2. The results are available at time $t_3 << t_8$.

Concurrency and communication

- Communication speed is considerably slower than computation speed. During the time to send or receive a few bytes a processor could execute billions of instructions.
- Intensive communication can slow down considerably the group of concurrent threads of an application.

Barrier synchronization

Barrier synchronization → sometimes, a computation consists of multiple stages when concurrently running threads cannot continue to the next stage until all of them have finished the current one. This leads to inefficiencies.



Barrier synchronization. Seven threads start execution of Stage 1 at time t_0. Threads T_1, T_3, T_4, T_5, T_6, and T_7 finish early and have to wait for thread T_2 before proceeding to Stage 2 at time t_1. Similarly, tasks T_1, T_2, T_3, T_4, T_6, and T_7 have to wait for task T_5, before proceeding to the next stage at time t_2. White bars represent blocked task, waiting to proceed to the next stage.

Parallel vs. distributed computing

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

- ***Parallel computing*** refers to a model in which the computation is divided among several processors sharing the same memory.
  - 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 the shared memory.
  - considered a single computer

Parallel vs. distributed computing

- ***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.
  - Therefore, includes a wider range of systems and applications than parallel computing and is often considered a more general term.
  - 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 are computing grids or Internet computing systems, which combine together the biggest variety of architectures, systems, and applications in the world.

Parallel Processing

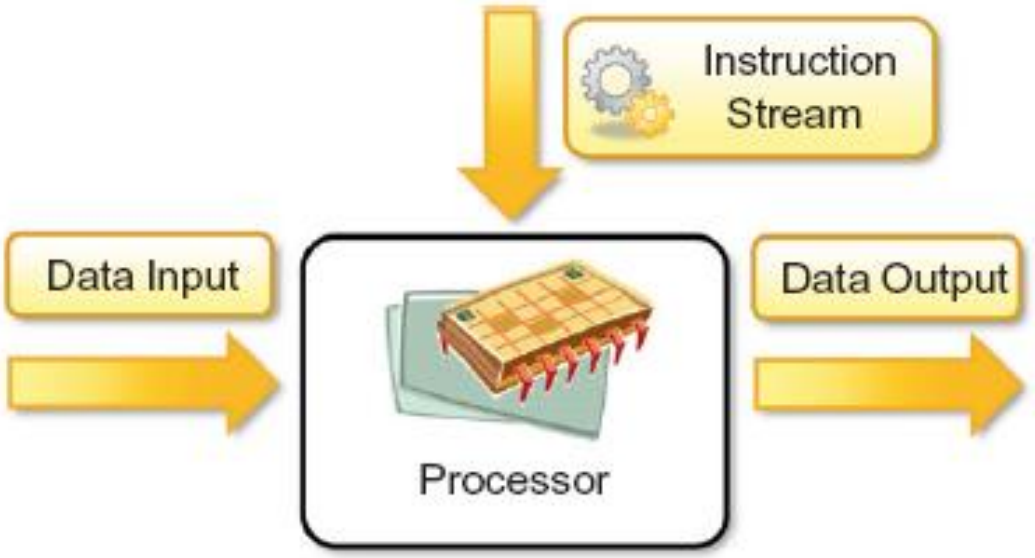
- Processing of multiple tasks simultaneously on multiple processors is called parallel processing.
- The parallel program consists of multiple active processes (tasks) simultaneously solving a given problem.
- A given task is divided into multiple subtasks using a divide-and-conquer technique, and each subtask is processed on a different central processing unit (CPU).
- Programming on a multiprocessor system using the divide-and-conquer technique is called parallel programming.

Hardware architectures for parallel processing

- 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

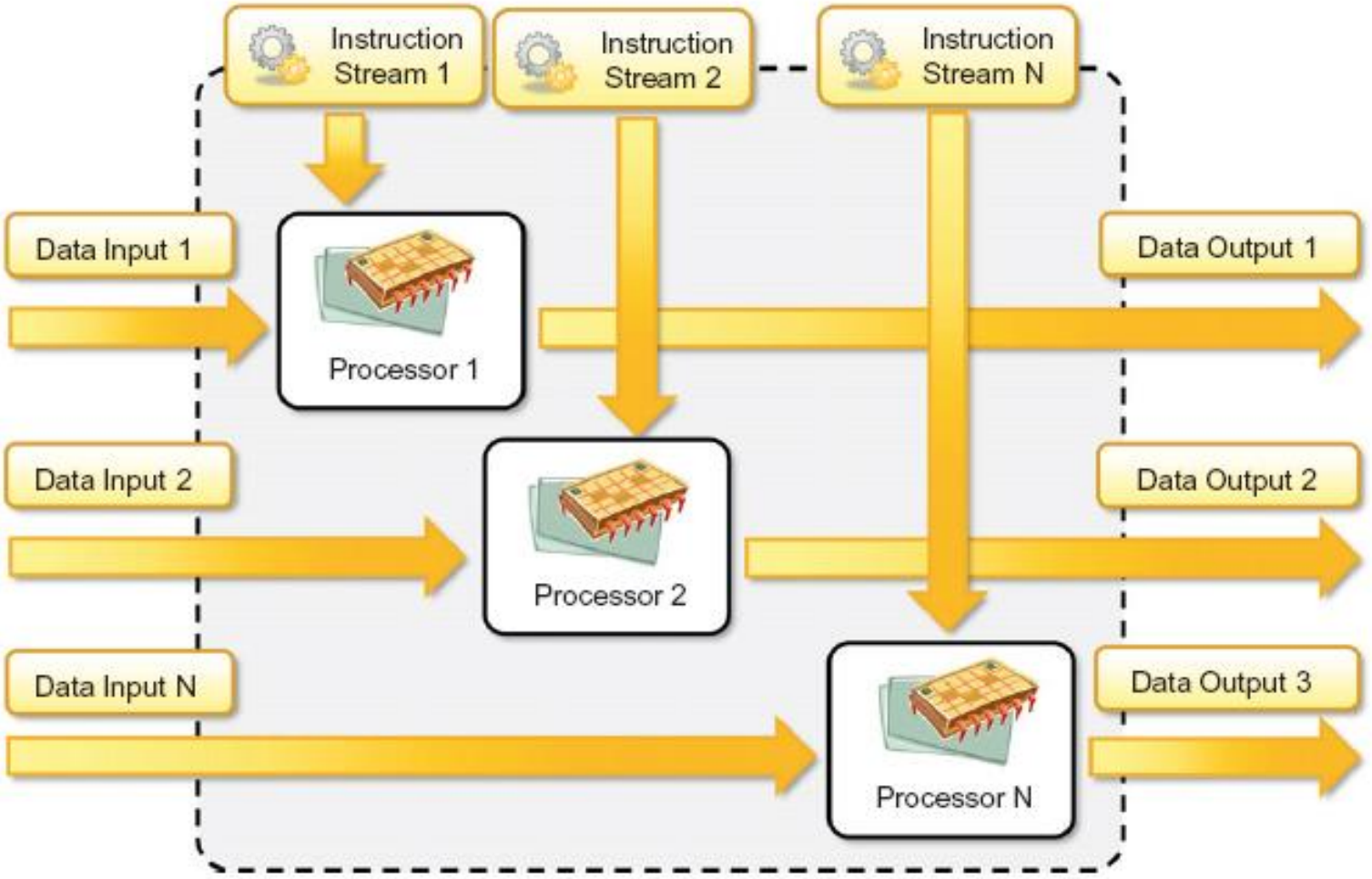
Single-instruction, single-data (SISD) systems

- 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; hence computers adopting this model are popularly called *sequential computers*.



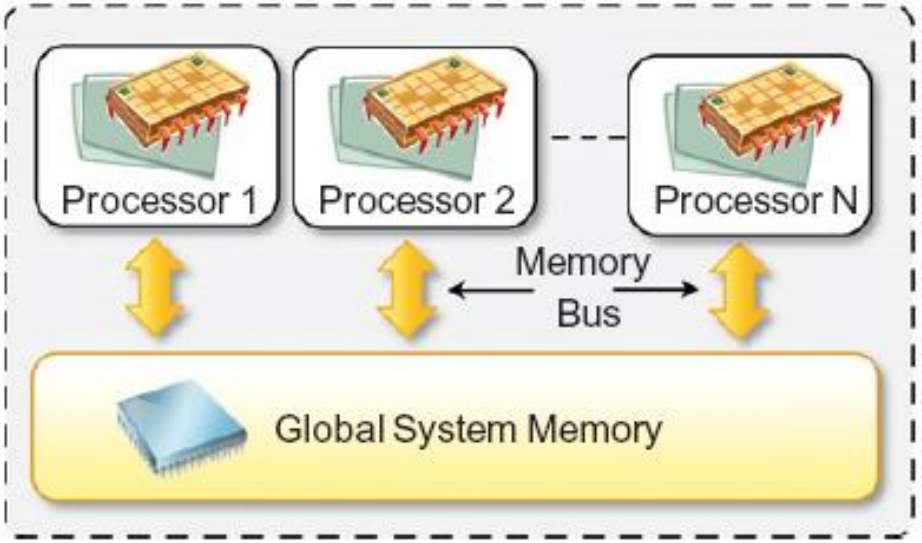
Multiple-instruction, multiple-data (MIMD) systems

- An MIMD computing system is a multiprocessor machine capable of executing multiple instructions on multiple data sets. Each Processing Element (PE) in the MIMD model has separate instruction and data streams; PEs in MIMD machines work asynchronously.



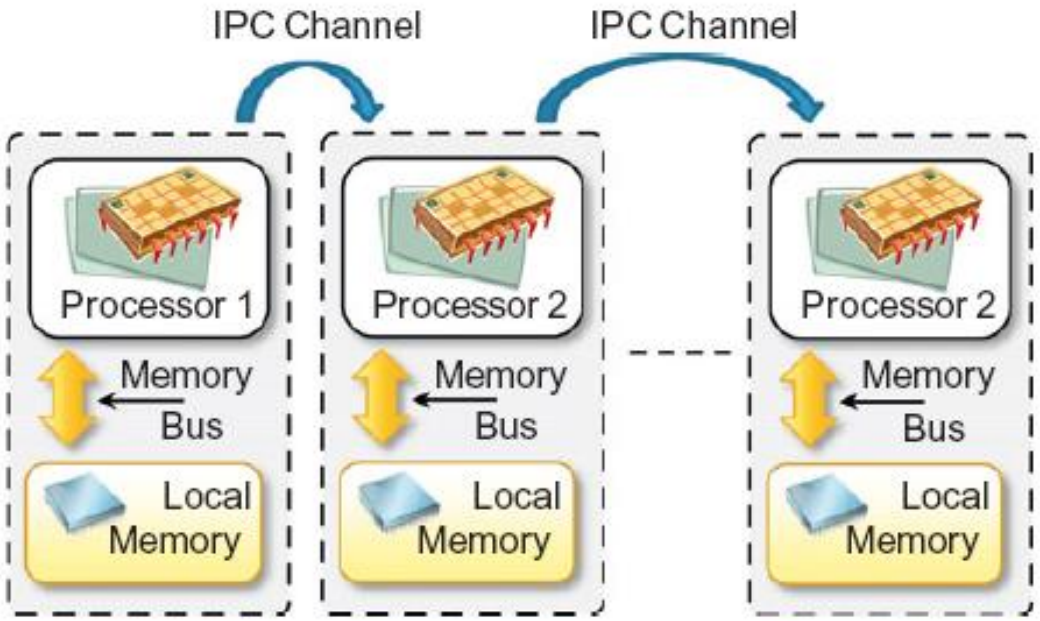
Shared memory MIMD machines

- 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 multiprocessor systems.
- The communication between PEs in this model takes place through the shared memory.
- Therefore, modification of the data stored in the global memory by one PE is visible to all other PEs. Easier to program but is less tolerant to failures.
- Dominant representative shared memory MIMD systems are Silicon Graphics machines and Sun/IBM's SMP (Symmetric Multi-Processing).

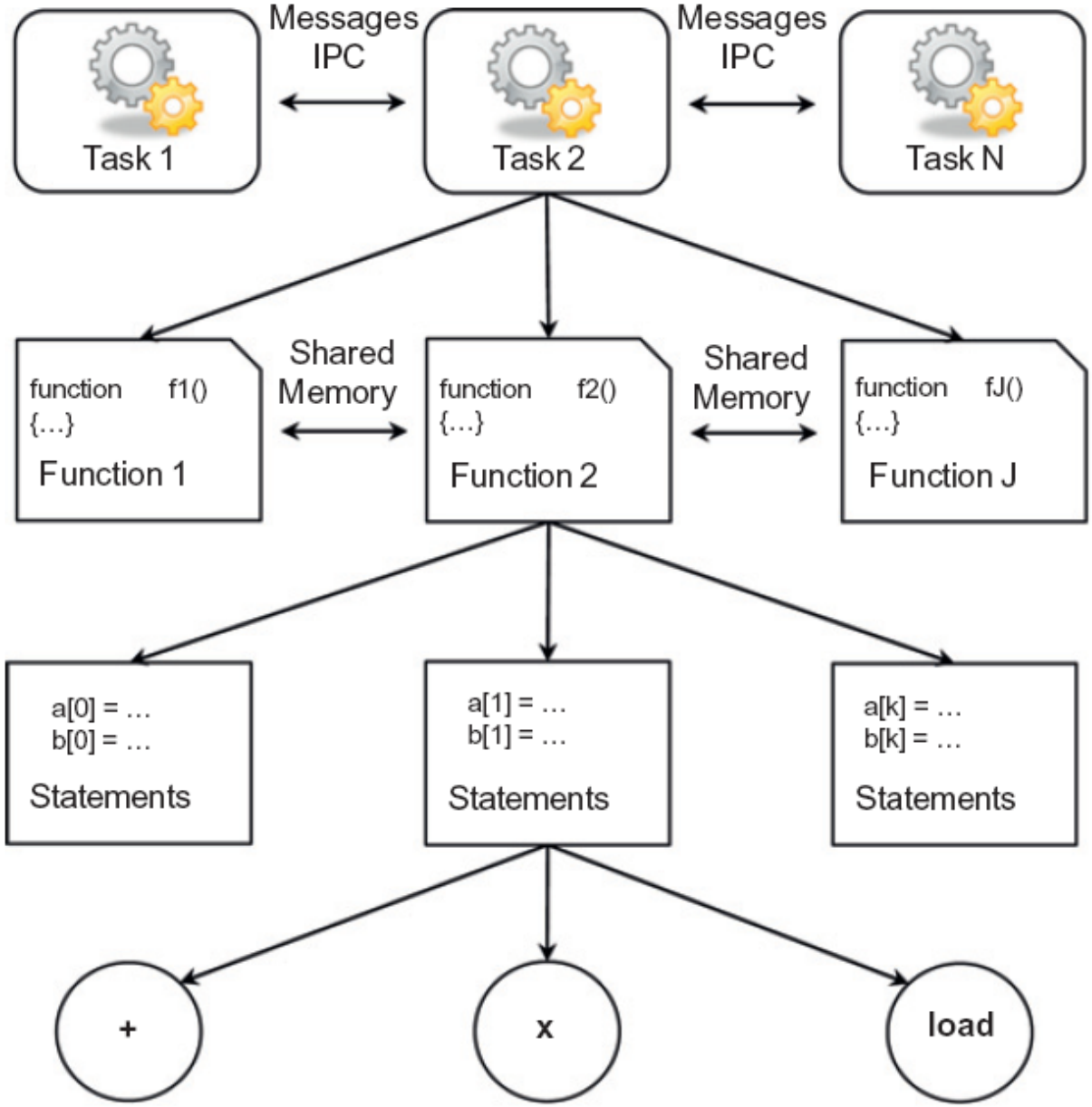


Distributed memory MIMD machines

- 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 interprocess communication channel, or IPC).
- Each PE operates asynchronously, and if communication/synchronization among tasks is necessary, they can do so by exchanging messages between them.
- 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.



Levels of parallelism



Programmer

Large Level
(Processes, Tasks)

Processes → Tasks

Programmer

Medium Level
(Threads, Functions)

Tasks → Threads, functions, procedures

Parallelizing compiler

Fine Level
(Processor,
Instructions)

Functions → Loops, instruction blocks

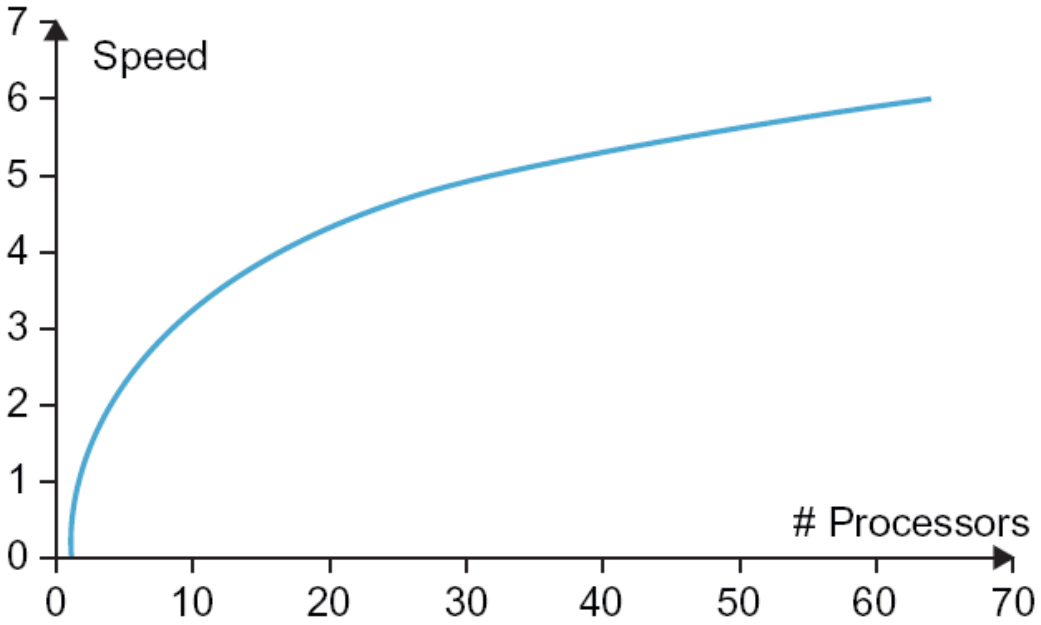
Processor

Very Fine Level
(Cores, Pipeline,
Instructions)

Loops → Instructions

Laws of caution

- 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 increased by n times. This is an ideal situation, but it rarely happens because of the communication overhead.
- Speed by a parallel computer increases as the logarithm of the number of processors (i.e., $y = k*log(N)$).



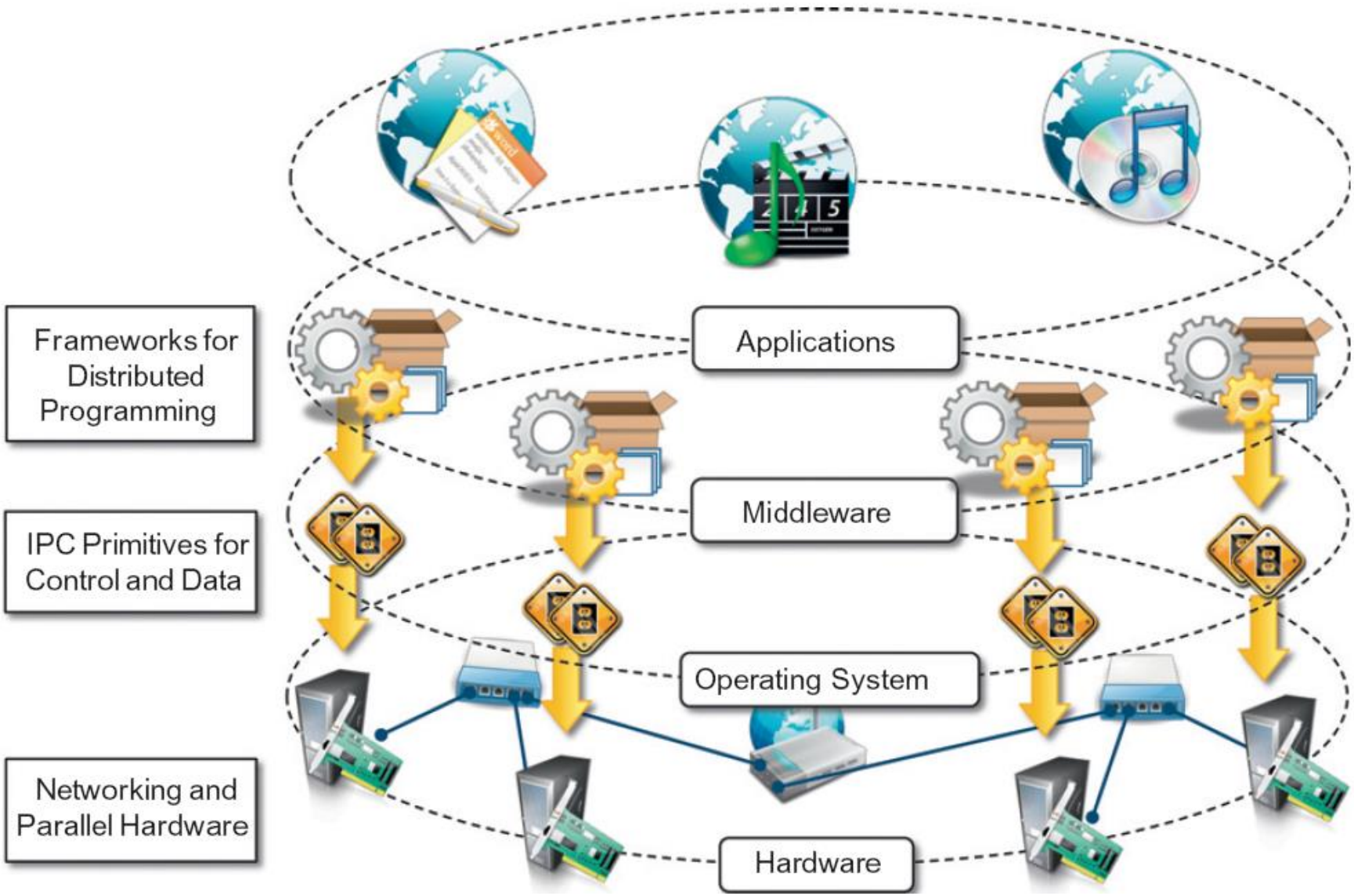
Components of a distributed system – Middleware

- 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.

- The middleware layer relying on the services offered by the operating system, the middleware develops its own protocols, data formats, and programming language or frameworks for the development of distributed applications.

- All of them constitute a uniform interface to distributed application developers that is completely independent from the underlying operating system and hides all the heterogeneities of the bottom layers.

Middleware (cont'd)

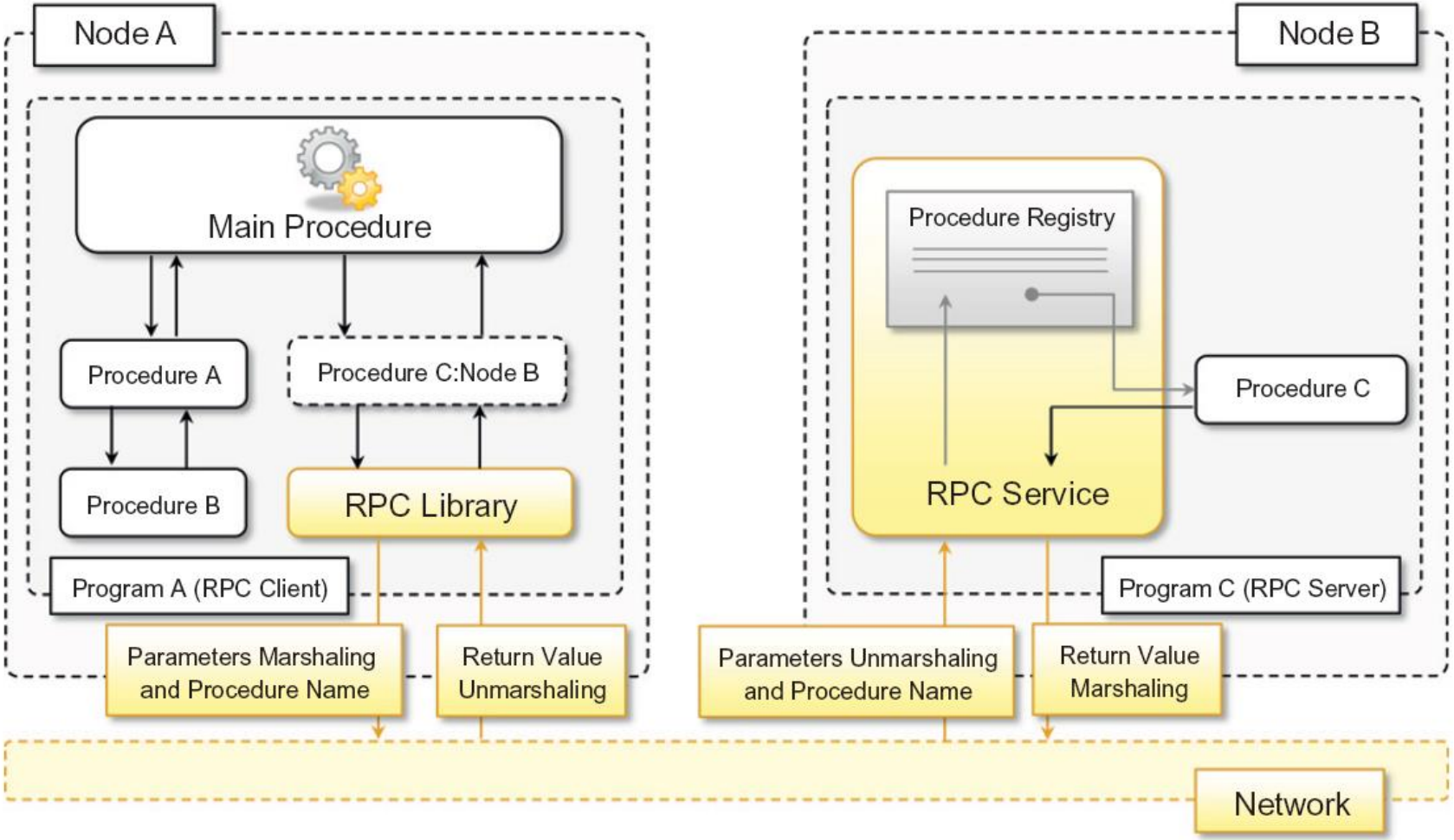


Models for interprocess communication (IPC)

- Distributed systems are composed of a collection of concurrent processes interacting with each other by means of a network connection. Therefore, 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 - most relevant that we can mention are shared memory, remote procedure call (RPC), and message passing.
- Sockets are the most popular IPC primitive for implementing communication channels between distributed processes.

Remote procedure call

- RPC has been a dominant technology for IPC for quite a long time.
- This paradigm extends the concept of procedure call beyond the boundaries of a single process, thus triggering the execution of code in remote processes.
- The called procedure and calling procedure may be on the same system or they may be on different systems. Therefore this allows objects to be distributed across a heterogeneous network.
- The important aspect of RPC is marshalling and unmarshalling (parameter and return values - transported over a network through a sequence of bytes).
- Several programming languages and environments support this interaction pattern in the form of libraries and additional packages.
  - For instance, RPyC is an RPC implementation for Python.
  - There also exist platform-independent solutions such as XML-RPC and JSON-RPC, which provide RPC facilities over XML and JSON, respectively.
  - Thrift is the framework developed at Facebook.
  - Currently, the term RPC implementations encompass a variety of solutions including frameworks such distributed object programming (CORBA, DCOM, Java RMI, and .NET Remoting).



- The system is based on a client/server model. The server process maintains a registry of all the available procedures that can be remotely invoked and listens for requests from clients that specify which procedure to invoke, together with the values of the parameters required by the procedure. RPC maintains the synchronous pattern that is natural in IPC and function calls. Therefore, the calling process thread remains blocked until the procedure on the server process has completed its execution and the result (if any) is returned to the client.

Distributed object frameworks

- 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.

- **Common object request broker architecture (CORBA):** CORBA is a specification introduced by the Object Management Group (OMG) for providing cross-platform and cross-language interoperability among distributed components.
- The current release of the CORBA specification is version 3.0 and currently the technology is not very popular, mostly because the development phase is a considerably complex task and the interoperability among components developed in different languages has never reached the proposed level of transparency.

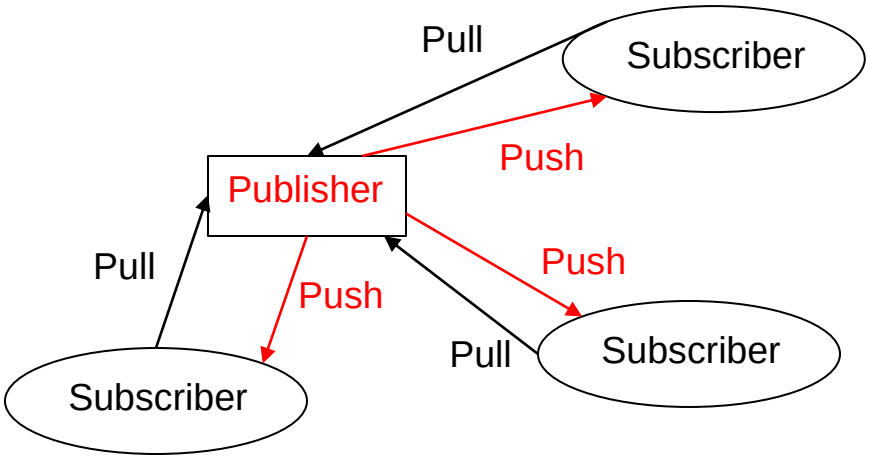
Examples of distributed object frameworks (cont'd)

- **Java remote method invocation (RMI):** Java RMI is a standard technology provided by Java for enabling RPC among distributed Java objects. RMI defines an infrastructure allowing the invocation of methods on objects that are located on different Java Virtual Machines (JVMs) residing either on the local node or on a remote one. Developers define an interface extending *java.rmi.Remote* that defines the contract for IPC.
- Once the development and deployment phases are completed and a reference to a remote object is obtained, the client code interacts with it as though it were a local instance, and RMI performs all the required operations to enable the IPC.

- **.NET remoting:** Remoting is the technology allowing for IPC among .NET applications. It provides developers with a uniform platform for accessing remote objects from within any application developed in any of the languages supported by .NET.
- Both technologies (Java RMI and .NET Remoting) have been extensively used to develop distributed applications.

Models for Message-based communication (cont'd)

- Point-to-Point message model (like a Request-Reply model) - Each message is sent from one component to another, and there is a direct addressing to identify the message receiver. Direct communication or queue-based communication.
- Publish – and – Subscribe message model
  - Push Strategy: responsibility of the publisher to notify all the subscribers.
  - Pull Strategy : responsibility of the subscribers to check whether there are messages on the events that are registered.



Models for Message-based communication (cont'd)

- Also message passing paradigm can be given as an example of message-based communication. Examples of this model are the Message-Passing Interface (MPI) and OpenMP.

- It is very uncommon that one single mode satisfies all the communication needs within a system. More likely, a composition of modes or their conjunct use in order to design and implement different aspects is the common case.

Service Oriented Architecture

- Service Oriented Architecture (SOA) is a well established architectural approach for designing and developing applications in the form services that can be shared and reused.
- A service-oriented application is generally composed of services that are spread across different domains, trust authorities, and execution environments.
- The services communicate with each other by passing messages.
- Services are described using the Web Services Description Language (WSDL).
- Even though there is no designed technology for the development of service-oriented software systems, Web services are the de facto approach for developing SOA. Web services, the fundamental component enabling cloud computing systems, leverage the Internet as the main interaction channel between users and the system.
- WSDL is an XML-based web services description language that is used to create service descriptions containing information on the functions performed by a service and the inputs and outputs of the service.

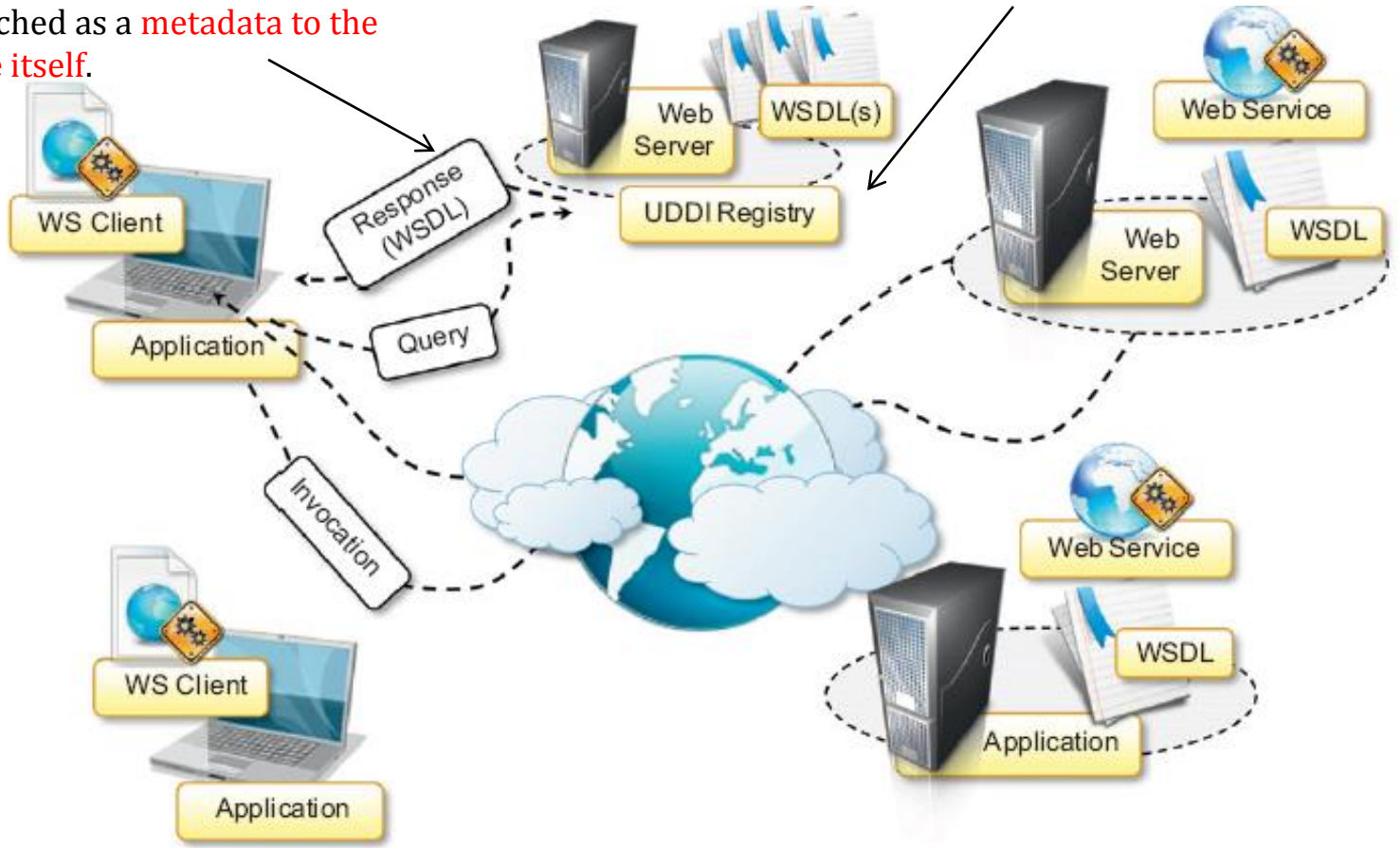
Web-services

- Web service technology provides an implementation of the RPC concept over HTTP, thus allowing the interaction of components that are developed with **different technologies (interoperability across different platforms and programming languages).**
- A Web service is exposed as a remote object hosted on a Web server, and method invocations are transformed in HTTP requests, opportunely packaged using specific protocols such as **Simple Object Access Protocol (SOAP)** or **Representational State Transfer (REST).**
- Several aspects make Web services the technology of choice for SOA.
- They are based on well-known and vendor-independent standards such as HTTP, SOAP, XML, and WSDL .
- The semantics for invoking Web service methods is expressed through interoperable standards such as XML and WSDL, which also provide a complete framework for expressing simple and complex types in a platform-independent manner.

Web-services (cont'd)

The service description document, expressed by means of Web Service Definition Language (WSDL), can be either uploaded to a global registry or attached as a metadata to the service itself.

Service consumers can look up and discover services in global catalogues (registry) using Universal Description Discovery and Integration (UDDI).



Web-services (cont'd)

- The backbone of all these technologies is XML, which is also one of the causes of Web services' popularity and ease of use.
- XML-based languages are used to manage the low-level interaction for Web service method calls (SOAP), for providing metadata about the services (WSDL), for discovery services (UDDI), and other core operations. In practice, the core components that enable Web services are SOAP and WSDL.
- Clients for web services can be generated for **any language** that is capable of **interpreting XML data**. This is a fundamental feature that enables Web service interoperability and one of the reasons that make such technology a solution of choice for SOA.

Web-services – SOAP messages (request)

```
POST /InStock HTTP/1.1
Host: www.stocks.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: <Size>

<?xml version="1.0">
<soap:Envelope xmlns:soap="http//www.w3.org/2001/12/soap-envelope"
 soap:encondingStyle="http//www.w3.org/2001/12/soap-encoding">
 <soap:Header></soap:Header>
 <soap:Body xmlns:m=http://www.stocks.org/stock>
  <m:GetStockPrice>
   <m:StockName>IBM<m:StockName>
  </m:GetStockPrice>
 </soap:Body>
</soap:Envelope>
```

The envelope defines the boundaries of the SOAP message.

Envelope

Header: Metadata & Assertions

Body: Method Call

The header is optional and contains relevant information on how to process the message. In addition, it contains information such as routing and delivery settings, authentication and authorization assertions, and transaction contexts.

The body contains the actual message to be processed.

Web-services – SOAP messages (reply)

```
POST /InStock HTTP/1.1
Host: www.stocks.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: <Size>

<?xml version="1.0">
<soap:Envelope xmlns:soap="http//www.w3.org/2001/12/soap-envelope"
 soap:encondingStyle="http//www.w3.org/2001/12/soap-encoding">
 <soap:Header></soap:Header>
 <soap:Body xmlns:m=http://www.stocks.org/stock>
  <m:GetStockPriceResponse>
   <m:Price>34.5<m:Price>
  </m:GetStockPriceResponse>
 </soap:Body>
</soap:Envelope>
```

Envelope

Header: Metadata & Assertions

SOAP message used to invoke a Web service method that retrieves the price of a given stock and the corresponding reply.

Body: Execution Result

GetStockPrice method and receiving the result **do not have any information** about the type and structure of the parameters and the return values. This information is stored within the WSDL document attached to the Web service.

Web-services (cont'd)

- SOAP has often been considered quite inefficient because of the excessive use of markup that XML imposes for organizing the information into a well-formed document. Therefore, lightweight alternatives to the SOAP/XML pair have been proposed to support Web services.
- The most relevant alternative is Representational State Transfer (REST), which provides a model for designing network-based software systems utilizing the client/server model and leverages the facilities provided by HTTP for IPC without additional burden.
- In a *RESTful* system, a client sends a request over HTTP using the standard HTTP methods (*PUT*, *GET*, *POST*, and *DELETE*), and the server issues a response that includes the representation of the resource.
- The content of data is still transmitted using XML as part of the HTTP content, but the additional markup required by SOAP is removed.
- Moreover, an alternatives to XML are *Asynchronous JavaScript and XML (AJAX)*, *JavaScript Standard Object Notation (JSON)*, which allows representing objects and collections of objects in a platform-independent manner.

Web-services – REST sample

REST JSON Sample Request

```
{"user":{"firstName":"John","lastName":"Smith","e
mail":"john.smith@pocahontas.com"}
```

REST JSON Sample Response

```
{"user":{"firstName":"John","lastName":"Smith","e
mail":"john.smith@pocahontas.com"},"_links":{"edi
t":["href","http:\/\/www.mysite.com\/api\/user\/1
0"],"message":["href","http:\/\/www.mysite.com\/a
pi\/user\/10\/message"]}}
```

From developer.constantcontract.com