Distributed Computing

Evaluation:

Final Exam	Mid	Lab	Total
75	15	10	100

Distributed Computing Syllabus

Chapter 1: Introduction

Chapter 2: A Model of Distributed Computations

Chapter 3: Logical Time

Chapter 4: Interconnection Networks (static & dynamic)

Chapter 5:Terminology and Basic Algorithms

Chapter 6: Parallel Programming Models

Chapter1 : Introduction

Distributed System

- Definition
  - A distributed system is collection of loosely coupled processors communicating over a communication network in order to solve a problem that is too hard or cannot be individually solved.
  - A computer program that runs in a distributed system is called a distributed program.

The main characteristics of distributed system are:

- No common physical clock.
- No shared memory.
- Geographical separation.
- Autonomy and heterogeneity.

Distributed System Model

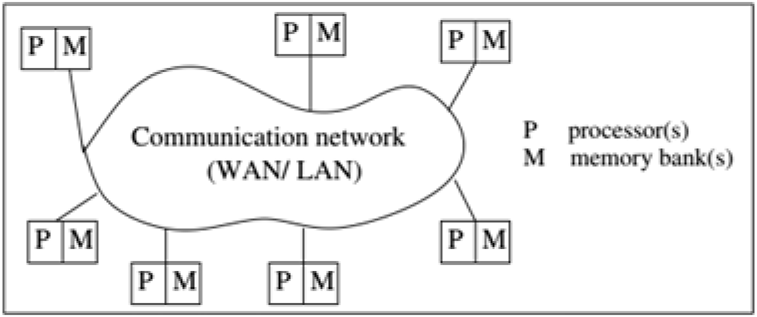


Figure 1.1: A distributed system connects processors by a communication network.

Relation between Software Components

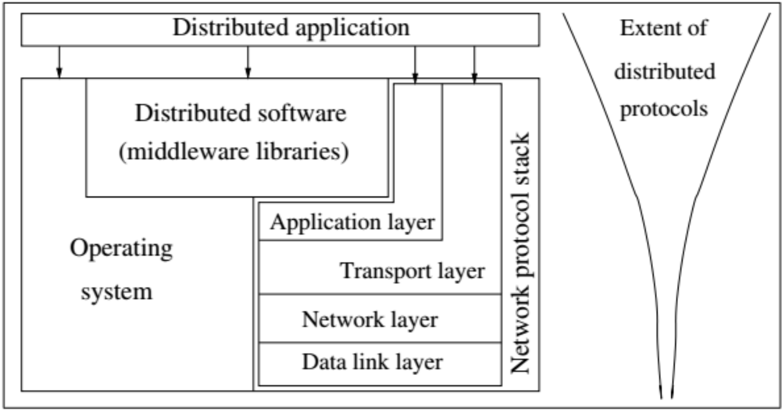


Figure 1.2: Interaction of the software components at each process.

Motivation for Distributed System

- Inherently distributed computation
- Resource sharing
- Access to remote resources
- Increased performance/cost ratio
- Reliability
  - ➢ availability, integrity, fault-tolerance
- Scalability
- Modularity and incremental expandability

Parallel Systems

- Multiprocessor systems (direct access to shared memory, UMA model)
  - ➢ each node (processor) is connected to each other by the network. The Interconnection network : bus, multi-stage switch.
  - ➢ The connection between each stage is through interconnection generation function, or routing function in case of switch interconnection.
  - ➢ Access latency to complete an access to any memory location is the same.
  - ➢ E.g., Omega, Butterfly, Clos, Shuffle - exchange networks

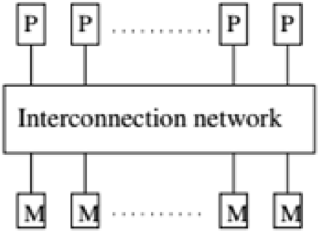


Figure 1.3.1: Standard architecture for parallel systems: Uniform memory access (UMA) multiprocessor system

M memory   P processor

UMA Model

Omega, Butterfly Interconnects

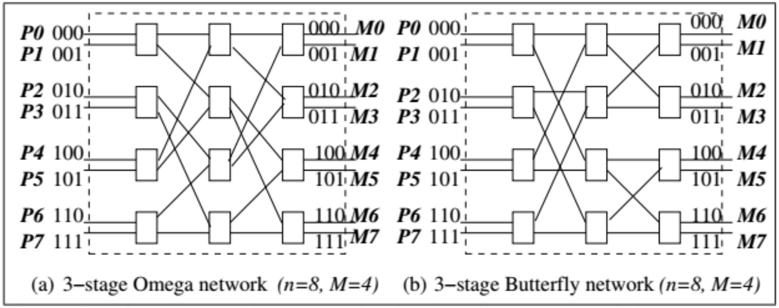


Figure 1.4: Interconnection networks for shared memory multiprocessor systems. (a) Omega network (b) Butterfly network.

Parallel Systems continued

•Multicomputer parallel systems (no direct access to shared memory, NUMA model)

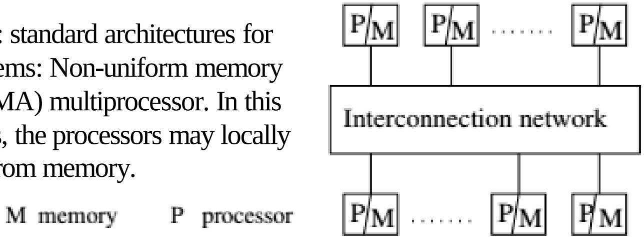
➢The NUMA architecture defines the node as the Processing Element, with a part of the main memory. Then, each node is connected to each other by the network.

➢access latency to various memory locations from different processors varies (Access to local memory takes less time than access to remote memory).

➢Bus, ring, mesh (wrap around), hypercube topologies

➢E.g., NYU Ultracomputer, CM* Conneciton Machine, IBM Blue gene

Figure 1.3.2: standard architectures for parallel systems: Non-uniform memory access (NUMA) multiprocessor. In this architectures, the processors may locally cache data from memory.



NUMA Model

Interconnection Topologies for Multiprocessors

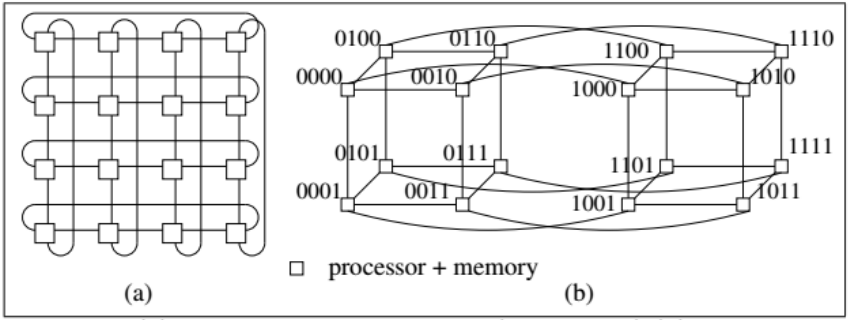


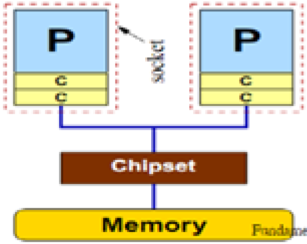
Figure 1.5: (a) 2-D Mesh with wraparound (a.k.a. torus) (b) 3-D hypercube

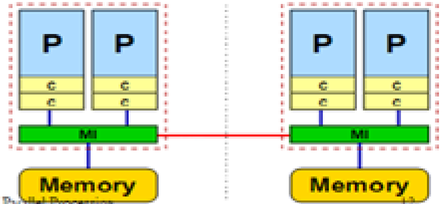
Uniform memory access(UMA)

- Identical processors
- Equal access and access times to memory
- Sometimes called CC-UMA - Cache Coherent UMA. Cache coherent means if one processor updates a location in shared memory, all the other processors know about the update. Cache coherency is accomplished at the hardware level.

Non-Uniform Memory Access

- Often made by physically linking two or more SMPs
- One SMP can directly access memory of another SMP
- Not all processors have equal access time to all memories
- Memory access across link is slower
- If cache coherency is maintained, then may also be called CC-NUMA - Cache Coherent NUMA





SMP (symmetric multiprocessing)
is the processing of programs by multiple processors
that share a common operating system and memory.
In symmetric (or "tightly coupled") multiprocessing,
the processors share memory and the I/O bus or data
path. A single copy of the operating system is in charge
of all the processors. SMP, also known as a "shared
everything" system, does not usually exceed 16
processors. SMP systems are considered better than
MPP systems for online transaction processing (OTP)
in which many users access the same database in
a relatively simple set of transactions. An advantage
of SMP for this purpose is the ability to dynamically
balance the workload among computers (and as a
result serve more users faster).

MPP (massively parallel processing) is the coordinated processing of a program by multiple processors that work on different parts of the program, with each processor using its own operating system and memory . Typically, MPP processors communicate using some messaging interface. In some implementations, up to 200 or more processors can work on the same application. An "interconnect" arrangement of data paths allows messages to be sent between processors. Typically, the setup for MPP is more complicated, requiring thought about how to partition a common database among processors and how to assign work among the processors. An MPP system is also known as a "loosely coupled" or "shared nothing" system.

An MPP system is considered better than a symmetrically parallel system (SMP) for applications that allow a number of databases to be searched in parallel. These include decision support system and data warehouse applications.

Flynn's Taxonomy (Processing models)

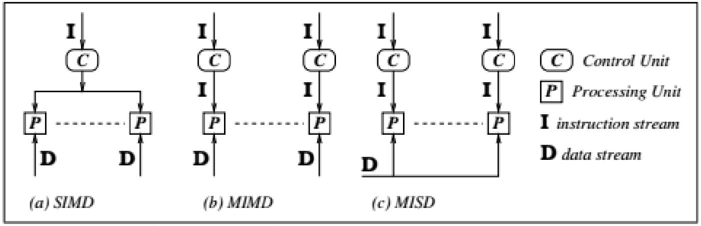
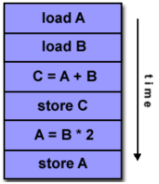


Figure 1.6: SIMD, MISD, and MIMD modes.

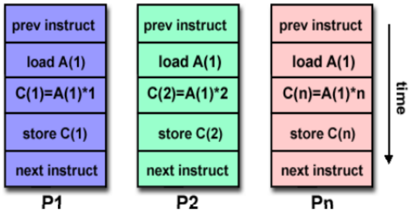
- SISD: Single Instruction Stream Single Data Stream (traditional)
- SIMD: Single Instruction Stream Multiple Data Stream
  - scientific applicaitons, applications on large arrays
  - vector processors, systolic arrays, Pentium/SSE, DSP chips
- MISD: Multiple Instruciton Stream Single Data Stream
  - E.g., visualization
- MIMD: Multiple Instruction Stream Multiple Data Stream
  - distributed systems, vast majority of parallel systems

Flynn's Classical Taxonomy

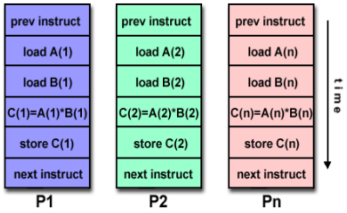
- SISD: A serial computer



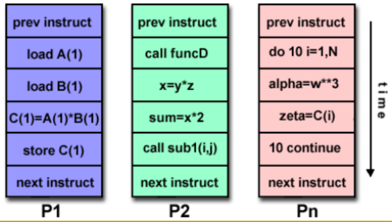
MISD: Cryptographic Decoding



- SIMD: GPUs



MIMD: Clusters, Supercomputers



Primitives for communication in distributed system

A distributed system communicate through message passing technique. In this technique Send and receive primitives are used.

- In **send primitive** (Send ())the data are copied from the user buffer to the kernel buffer. The data later gets copied from the kernel buffer onto the network.

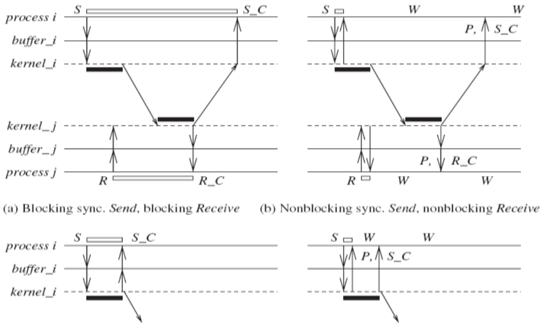
- In **receive primitive** (Receive()) the data is copied into kernel buffer and then copied from kernel buffer to user buffer.

- Synchronous (send/receive)
  - Handshake between sender and receiver
  - Send completes when Receive completes
  - Receive completes when data copied into buffer
- Asynchronous (send)
  - Control returns to process when data copied out of user-specified buffer

- Blocking (send/receive)
  - Control returns to invoking process after processing of primitive (whether sync or async) completes
- Nonblocking (send/receive)
  - Control returns to process immediately after invocation
  - Send: even before data copied out of user buffer
  - Receive: even before data may have arrived from sender

Blocking/non-blocking, synchronous/ asynchronous primitives

Figure 1.8 Blocking/ non-blocking and synchronous/asynchronous primitives [12]. Process P_i is sending and process P_j is receiving. (a) Blocking synchronous *Send* and blocking (synchronous) *Receive*. (b) Non-blocking synchronous *Send* and nonblocking (synchronous) *Receive*. (c) Blocking asynchronous *Send*. (d) Non-blocking asynchronous *Send*.



(a) Blocking sync. *Send*, blocking *Receive* (b) Nonblocking sync. *Send*, nonblocking *Receive*

(c) Blocking async. *Send* (d) Non-blocking async. *Send*

▬	Duration to copy data from or to user buffer		
▭	Duration in which the process issuing send or receive primitive is blocked		
S	*Send* primitive issued	*S_C*	processing for *Send* completes
R	*Receive* primitive issued	*R_C*	processing for *Receive* completes
P	The completion of the previously initiated nonblocking operation		
W	Process may issue *Wait* to check completion of nonblocking operation		

Asynchronous Executions in Message - Passing System

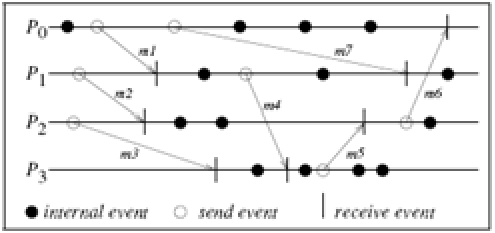
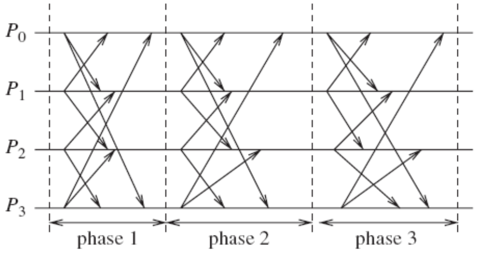


Figure 1.9 Asynchronous Executions in Message - Passing System

Synchronous Executions in Message - Passing System

Figure 1.10 An example of a synchronous execution in a message-passing system. All the messages sent in a round are received within that same round.



```
(1) $Sync\_Execution$(int $k$, $n$) //$k$ rounds, $n$ processes.
(2)   for $r = 1$ to $k$ do
(3)       proc $i$ sends msg to $(i+1)$ $mod$ $n$ and $(i-1)$ $mod$ $n$;
(4)       each proc $i$ receives msg from $(i+1)$ $mod$ $n$ and $(i-1)$ $mod$ $n$;
(5)       compute app-specific function on received values.
```

Asynchronous Executions

(i) There is no processor synchrony and there is no bound on the drift rate of processor clocks.

(ii) Message delays (transmission + propagation times) are finite but unbounded.

(iii) There is no upper bound on the time taken by a process to execute a step.

Synchronous Executions

(i) Processors are synchronized and the clock drift rate between any two processors is bounded.

(ii) Message delivery (transmission + delivery) times are such that they occur in one logical step or round.

(iii) There is a known upper bound on the time taken by a process to execute a step.

Terminology

Coupling: The degree of coupling among a set of modules, whether hardware or software, is measured in terms of the interdependency and binding and/or homogeneity among the modules. When the degree of coupling is high (low), the modules are said to be tightly (loosely) coupled

- **Tightly coupled** multiprocessors shared memory have the same type , speed (homogenous) ,the same operating system and very close to each other, e.g UMA multiprocessors and NUMA multicomputers.

- **Loosely coupled** multicomputers without shared memory and without common clock , heterogeneous and are physically remote.

Transparent system

A distributed system that is able to present itself to users and applications as if it were only a single computer system is said to be transparent.

Types of transparency

1- Access transparency

Hiding differences in data representation and the way that resources can be accessed by users.

2- Location transparency

Refers to the fact users can not tell where a resource is physically located in the system. Location transparency can be achieved by assigning only logical names to resources. e.g.URL http://www.prenhall.com

Transparent system cont.

3- Migration transparency
Refers to the resources can be moved without affecting how these resources can be accessed.

4- Relocation transparency
Is the situation in which resources can be relocated while they are being accessed without the user or application noticing any thing. E.g. is when mobile users can continue to use wireless laptops while moving from place to place without disconnected.

5-Replaction transparency
Deals with hiding the fact several copies of resource exist , replicas have the same name.

Transparent system cont.

6- Concurrency transparency
Hide that a resource may be shared by several competitive users

7- Failure transparency
Hide the failure and recovery of a resource.

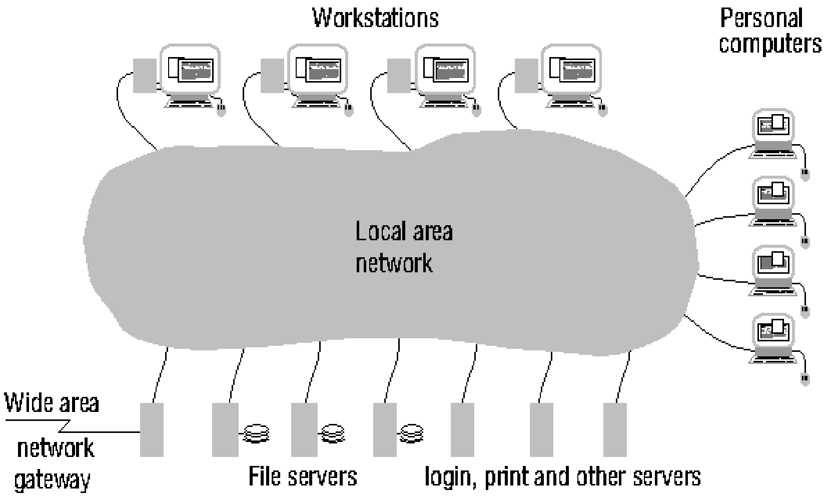
8. Scalability Transparency
Allows the system and applications to expand in scale without change to the system structure or the application algorithms.

Example: World-Wide-Web
Example: Distributed Database

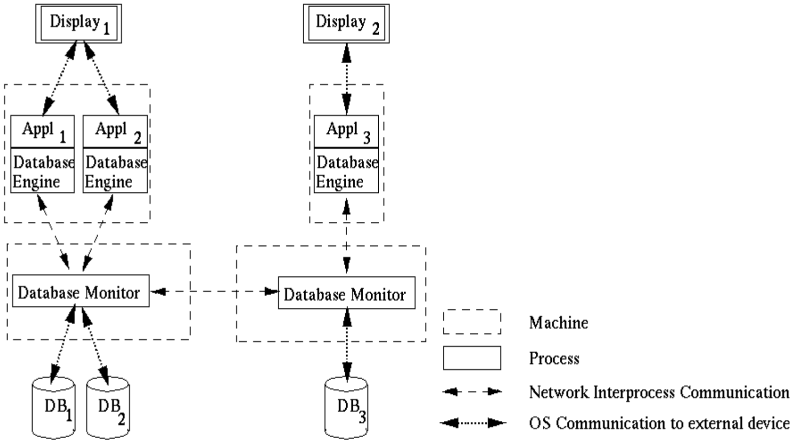
Examples of Distributed Systems

- Local Area Network and Intranet
- Database Management System
- Automatic Teller Machine Network
- Internet/World-Wide Web
- Mobile and Ubiquitous Computing

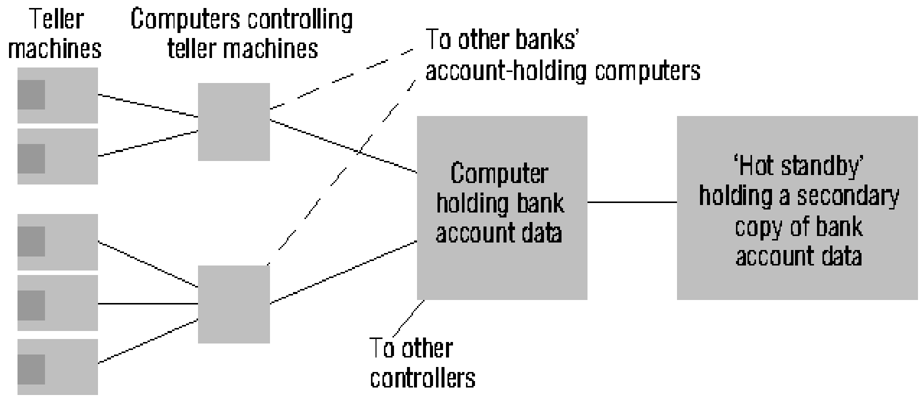
Local Area Network



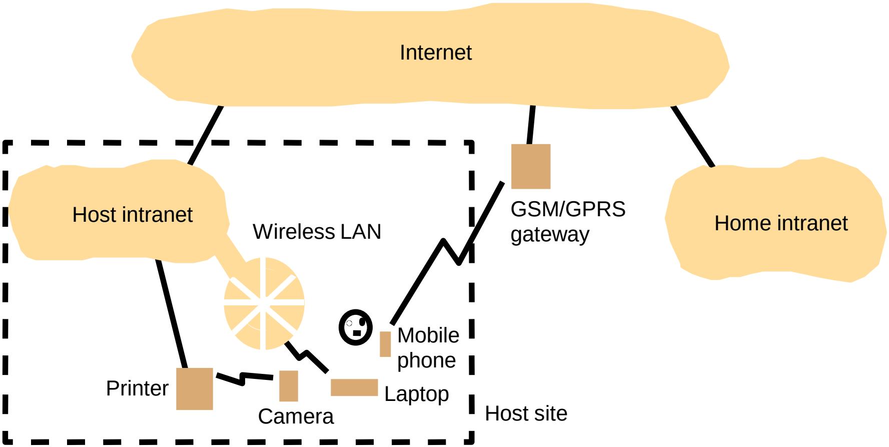
Database Management System



Automatic Teller Machine Network


Teller machines
Computers controlling teller machines
To other banks' account-holding computers
Computer holding bank account data
'Hot standby' holding a secondary copy of bank account data
To other controllers

Mobile and Ubiquitous Computing


Internet
Host intranet
Wireless LAN
GSM/GPRS gateway
Home intranet
Mobile phone
Printer
Camera
Laptop
Host site