MESSAGE-PASSING SYSTEMS VERSUS SHARED MEMORY SYSTEMS

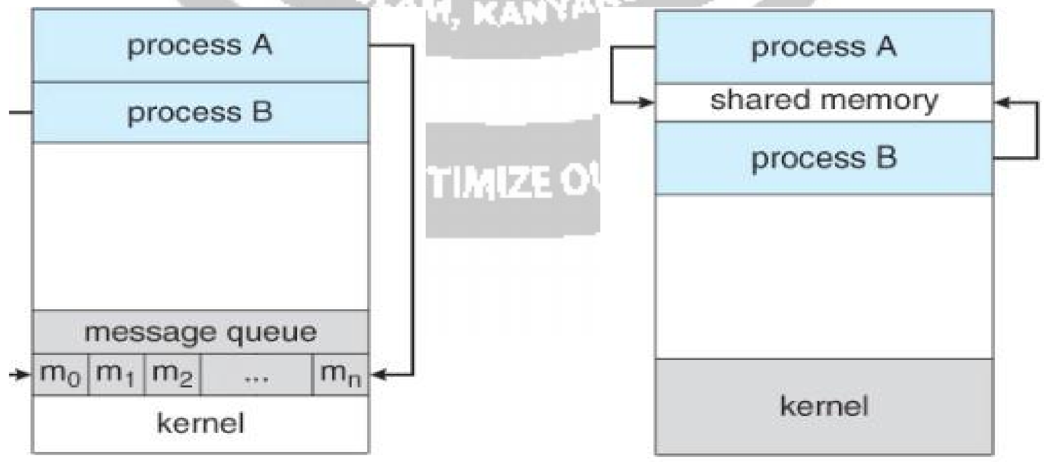
Communication among processors takes place via shared data variables, and control variables for synchronization among the processors**.** The communications between the tasks in multiprocessor systems take place through two main modes:

Message passing systems:

- This allows multiple processes to read and write data to the message queue without being connected to each other.
- Messages are stored on the queue until their recipient retrieves them. Message queues are quite useful for inter process communication and are used by most operating systems.

Shared memory systems:

- The shared memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other.
- Communication among processors takes place through shared data variables, and control variables for synchronization among the processors.
- Semaphores and monitors are common synchronization mechanisms on shared memory systems.
- When shared memory model is implemented in a distributed environment, it is termed as **distributed shared memory.**



a) Message Passing Model **b) Shared Memory Model**

Fig : Inter-process communication models

Differences between message passing and shared memory models

Message Passing	Distributed Shared Memory
Services Offered: Variables have to be marshalled from one process, transmitted and unmarshalled into other variables at the receiving process.	The processes share variables directly, so no marshalling and unmarshalling. Shared variables can be named, stored and accessed in DSM.
Processes can communicate with other processes. They can be protected from one another by having private address spaces.	Here, a process does not have private address space. So one process can alter the execution of other.
This technique can be used in heterogeneous computers.	This cannot be used to heterogeneous computers.
Synchronization between processes is through message passing primitives.	Synchronization is through locks and semaphores.
Processes communicating via message passing must execute at the same time.	Processes communicating through DSM may execute with non-overlapping lifetimes.
Efficiency: All remote data accesses are explicit and therefore the programmer is always aware of whether a particular operation is in-process or involves the expense of communication.	Any particular read or update may or may not involve communication by the underlying runtime support.

Emulating message-passing on a shared memory system (MP $\rightarrow$ SM)

- The shared memory system can be made to act as message passing system. The shared address space can be partitioned into disjoint parts, one part being assigned to each processor.
- Send and receive operations care implemented by writing to and reading from the destination/sender processor's address space. The read and write operations are synchronized.
- Specifically, a separate location can be reserved as the mailbox for each ordered pair of processes.

Emulating shared memory on a message-passing system (SM → MP)

- This is also implemented through read and write operations. Each shared location can be modeled as a separate process. Write to a shared location is emulated by sending an update message tothe corresponding owner process and read operation to a shared location is emulated by sending a query message to the owner process.
- This emulation is expensive as the processes have to gain access to other process memory location. The latencies involved in read and write operations may be high even when using shared memory emulation because the read and write operations are implemented by using network-wide communication.