Parallel Random Access Machines (PRAM) is a model, which is considered for most of the parallel algorithms. Here, multiple processors are attached to a single block of memory. A PRAM model contains—

- A set of similar type of processors.
- All the processors share a common memory unit. Processors can communicate among themselves through the shared memory only.
- A memory access unit (MAU) connects the processors with the single shared memory.

Here, \( n \) number of processors can perform independent operations on \( n \) number of data in a particular unit of time. This may result in simultaneous access of same memory location by different processors.

To solve this problem, the following constraints have been enforced on PRAM model—

- **Exclusive Read Exclusive Write (EREW)** — Here no two processors are allowed to read from or write to the same memory location at the same time.
- **Exclusive Read Concurrent Write (ERCW)** — Here no two processors are allowed to read from the same memory location at the same time, but are allowed to write to the same memory location at the same time.
- **Concurrent Read Exclusive Write (CREW)** — Here all the processors are allowed to write to the same memory location at the same time, but are not allowed to read from the same memory location at the same time.
- **Concurrent Read Concurrent Write (CRCW)** — All the processors are allowed to read from or write to the same memory location at the same time.

There are many methods to implement the PRAM model, but the most prominent ones are—

- Shared memory model
- Message passing model
- Data parallel model

**Shared Memory Model**
Shared memory emphasizes on **control parallelism** than on **data parallelism**. In the shared memory model, multiple processes execute on different processors independently, but they share a common memory space. Due to any processor activity, if there is any change in any memory location, it is visible to the rest of the processors.

As multiple processors access the same memory location, it may happen that at any particular point of time, more than one processor is accessing the same memory location. Suppose one is reading that location and the other is writing on that location. It may create confusion. To avoid this, some control mechanism, like **lock / semaphore**, is implemented to ensure mutual exclusion.

Shared memory programming has been implemented in the following –

- **Thread libraries** – The thread library allows multiple threads of control that run concurrently in the same memory location. Thread library provides an interface that supports multithreading through a library of subroutine. It contains subroutines for
  - Creating and destroying threads
  - Scheduling execution of thread
  - passing data and message between threads
  - saving and restoring thread contexts

Examples of thread libraries include – SolarisTM threads for Solaris, POSIX threads as implemented in Linux, Win32 threads available in Windows NT and Windows 2000, and JavaTM threads as part of the standard JavaTM Development Kit JDK.

- **Distributed Shared Memory DSM Systems** – DSM systems create an abstraction of shared memory on loosely coupled architecture in order to implement shared memory programming without hardware support. They implement standard libraries and use the advanced user-level memory management features present in modern operating systems. Examples include Tread Marks System, Munin, IVY, Shasta, Brazos, and Cashmere.

- **Program Annotation Packages** – This is implemented on the architectures having uniform memory access characteristics. The most notable example of program annotation packages is OpenMP. OpenMP implements functional parallelism. It mainly focuses on parallelization of loops.

The concept of shared memory provides a low-level control of shared memory system, but it tends to be tedious and erroneous. It is more applicable for system programming than application programming.

### Merits of Shared Memory Programming

- Global address space gives a user-friendly programming approach to memory.
- Due to the closeness of memory to CPU, data sharing among processes is fast and uniform.
- There is no need to specify distinctly the communication of data among processes.
- Process-communication overhead is negligible.
- It is very easy to learn.

**Demerits of Shared Memory Programming**

- It is not portable.
- Managing data locality is very difficult.

**Message Passing Model**

Message passing is the most commonly used parallel programming approach in distributed memory systems. Here, the programmer has to determine the parallelism. In this model, all the processors have their own local memory unit and they exchange data through a communication network.

![Diagram of Memory Units and Processors](image)

Processors use message-passing libraries for communication among themselves. Along with the data being sent, the message contains the following components −

- The address of the processor from which the message is being sent;
- Starting address of the memory location of the data in the sending processor;
- Data type of the sending data;
- Data size of the sending data;
- The address of the processor to which the message is being sent;
- Starting address of the memory location for the data in the receiving processor.

Processors can communicate with each other by any of the following methods −

- Point-to-Point Communication
- Collective Communication
- Message Passing Interface

**Point-to-Point Communication**

Point-to-point communication is the simplest form of message passing. Here, a message can be sent from the sending processor to a receiving processor by any of the following transfer modes −

- **Synchronous mode** – The next message is sent only after the receiving a confirmation that its previous message has been delivered, to maintain the sequence of the message.
- **Asynchronous mode** – To send the next message, receipt of the confirmation of the delivery of the previous message is not required.

**Collective Communication**

Collective communication involves more than two processors for message passing. Following
modes allow collective communications –

- **Barrier** – Barrier mode is possible if all the processors included in the communications run a particular block (known as barrier block) for message passing.

- **Broadcast** – Broadcasting is of two types –
  - **One-to-all** – Here, one processor with a single operation sends same message to all other processors.
  - **All-to-all** – Here, all processors send message to all other processors.

Messages broadcasted may be of three types –

- **Personalized** – Unique messages are sent to all other destination processors.
- **Non-personalized** – All the destination processors receive the same message.
- **Reduction** – In reduction broadcasting, one processor of the group collects all the messages from all other processors in the group and combine them to a single message which all other processors in the group can access.

**Merits of Message Passing**

- Provides low-level control of parallelism;
- It is portable;
- Less error prone;
- Less overhead in parallel synchronization and data distribution.

**Demerits of Message Passing**

- As compared to parallel shared-memory code, message-passing code generally needs more software overhead.

**Message Passing Libraries**

There are many message-passing libraries. Here, we will discuss two of the most-used message-passing libraries –

- Message Passing Interface **MPI**
- Parallel Virtual Machine **PVM**

**Message Passing Interface **MPI**

It is a universal standard to provide communication among all the concurrent processes in a distributed memory system. Most of the commonly used parallel computing platforms provide at least one implementation of message passing interface. It has been implemented as the collection of predefined functions called library and can be called from languages such as C, C++, Fortran, etc. MPIs are both fast and portable as compared to the other message passing libraries.

**Merits of Message Passing Interface**

- Runs only on shared memory architectures or distributed memory architectures;
- Each processors has its own local variables;
- As compared to large shared memory computers, distributed memory computers are less expensive.

**Demerits of Message Passing Interface**

- More programming changes are required for parallel algorithm;
Sometimes difficult to debug; and
Does not perform well in the communication network between the nodes.

**Parallel Virtual Machine PVM**

PVM is a portable message passing system, designed to connect separate heterogeneous host machines to form a single virtual machine. It is a single manageable parallel computing resource. Large computational problems like superconductivity studies, molecular dynamics simulations, and matrix algorithms can be solved more cost effectively by using the memory and the aggregate power of many computers. It manages all message routing, data conversion, task scheduling in the network of incompatible computer architectures.

**Features of PVM**

- Very easy to install and configure;
- Multiple users can use PVM at the same time;
- One user can execute multiple applications;
- It’s a small package;
- Supports C, C++, Fortran;
- For a given run of a PVM program, users can select the group of machines;
- It is a message-passing model,
- Process-based computation;
- Supports heterogeneous architecture.

**Data Parallel Programming**

The major focus of data parallel programming model is on performing operations on a data set simultaneously. The data set is organized into some structure like an array, hypercube, etc. Processors perform operations collectively on the same data structure. Each task is performed on a different partition of the same data structure.

It is restrictive, as not all the algorithms can be specified in terms of data parallelism. This is the reason why data parallelism is not universal.

Data parallel languages help to specify the data decomposition and mapping to the processors. It also includes data distribution statements that allow the programmer to have control on data – for example, which data will go on which processor – to reduce the amount of communication within the processors.