High Performance Computing

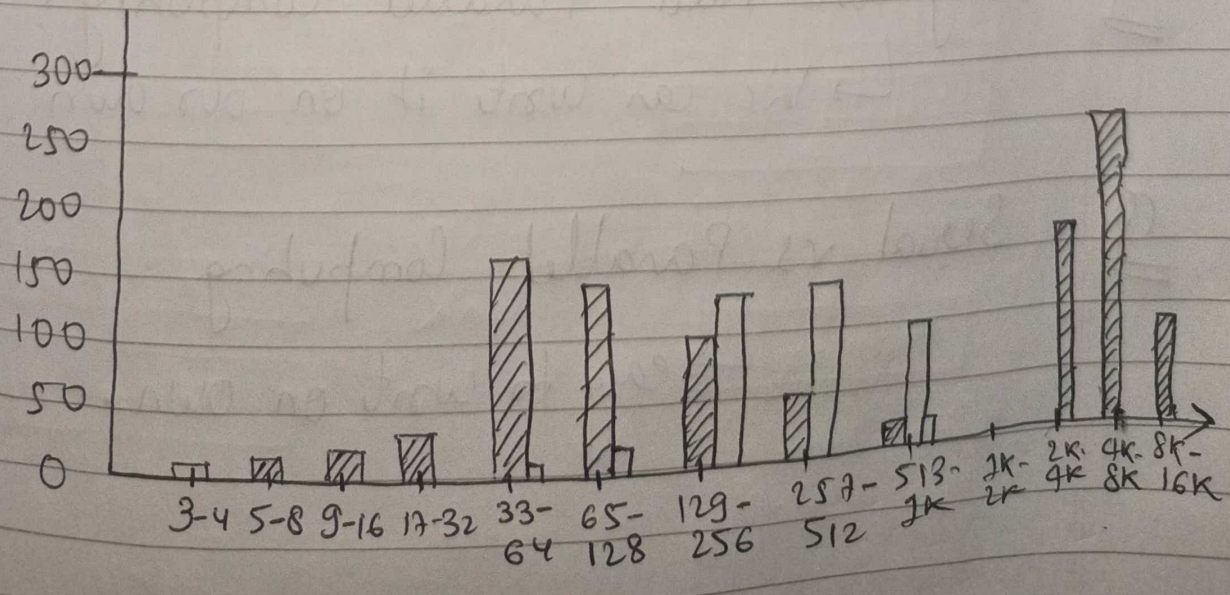
Unit-3

Parallel Computing :-

Parallel Computing refers to the process of breaking down larger problem into smaller, independent parts that can be executed simultaneously by multiple processors.

Adv :- 1) Saves time & money
2) CPU will not remain Idle.
3) It can solve impractical Serial computing problems.

Disadv :- 1) Algorithm will be complex
2) Low ~~Coml~~ coupling & High cohesion (required)
3) Sometimes it does not provides much benefit compared to the cost.



Application :-

1) Data bases and Data mining
2) Real time simulation of system
3) Science and Engineering
4) Advanced graphics
5) Augmented reality & virtual reality

In SuperComputers, LINPACK Benchmark is used to measure performance of super computer.

LINPACK Benchmark are a measure of a system floating point computing power (FLOPS). It measure how fast a system solves a complex & dense problem.

It is not generally accepted as a good metric because it covers only a single architectural aspect (peak performance).

Q. Why we need Parallel Computing?
↳ We can write it on our own

Q. Serial vs Parallel Computing
↳ easy to writ on own.

Ans → 1) Following are the various types of parallelism that can be used to compute large quantity of data and instruction of a computer :-

(i) Instruction level Parallelism :-

A processor can only address less than one instruction for each clock cycle phase. These instruction can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program. This is called instruction level parallelism.

(ii) Task-level parallelism :-

In this basically, a huge task is subdivided into small tasks and solve the problem concurrently.

(iii) Data-level parallelism :- In this, instruction from a single stream operate concurrently on multiple data at the same time.

(iv) Bit-level parallelism :-

It is form of parallelism which is based on increasing processor size. It reduces the number of instruction that the system must execute in order to perform a task on large sized data. For example, If a 16 bit instruction want to gets ~~executed~~ executed in a 8 bit system then first 8 bit get executed and then next 8 bit gets executed.

Flynn's Classification :-

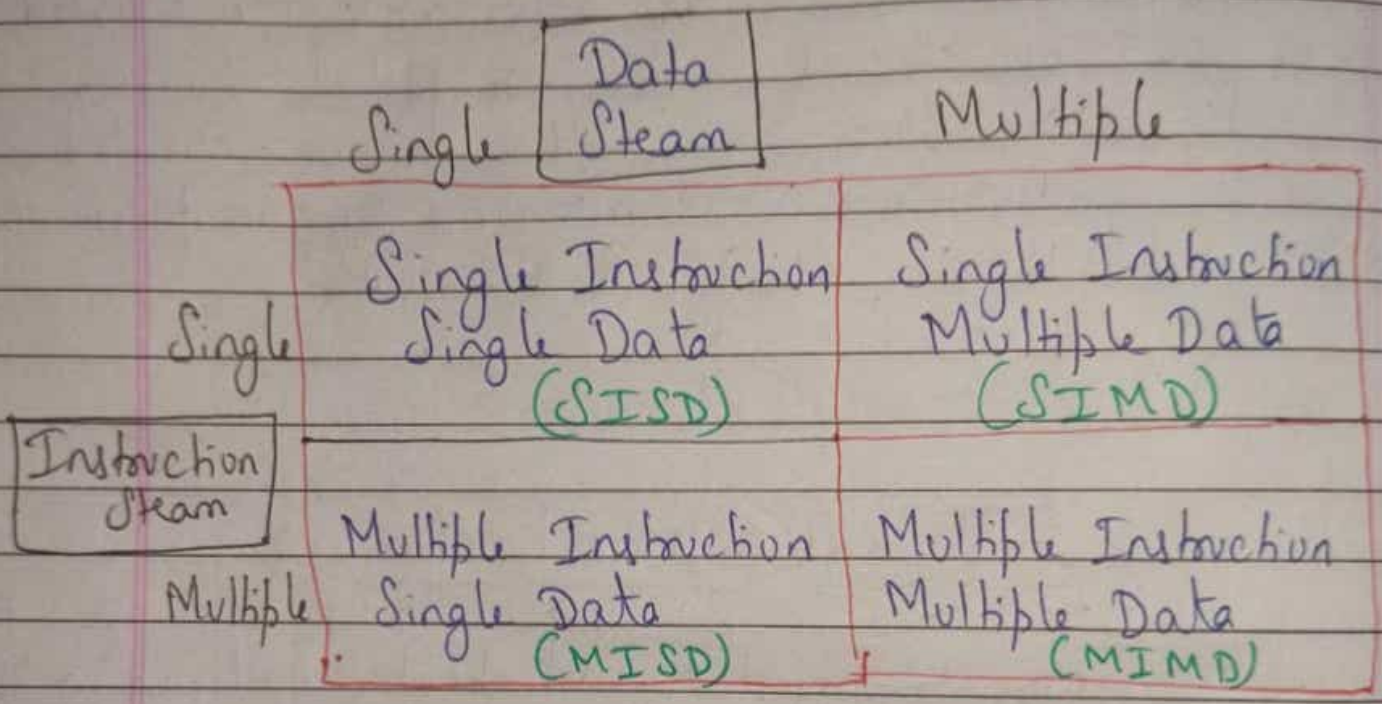
It was proposed by Micheal J. Flynn in 1966. In this classification computer are classified by whether it processes a single instruction at a time or multiple instruction simultaneously and whether it operate on one or multiple data sets.

This taxonomy distinguishes multiprocessor computer architectures according two independent dimension.

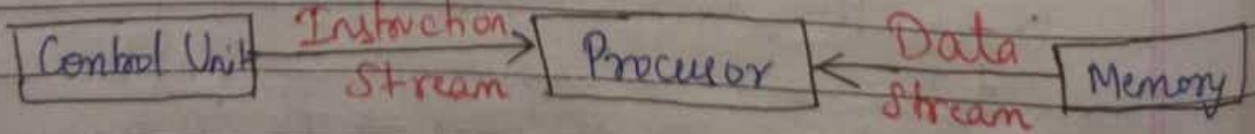
(a) Instruction steam → It is sequence of instruction executed by machine

(b) Data Steam → It is sequence of data including input, partial or temporary results used by instruction steam.

Each of these dimension can have only one of two possible states :- Single or Multiple.

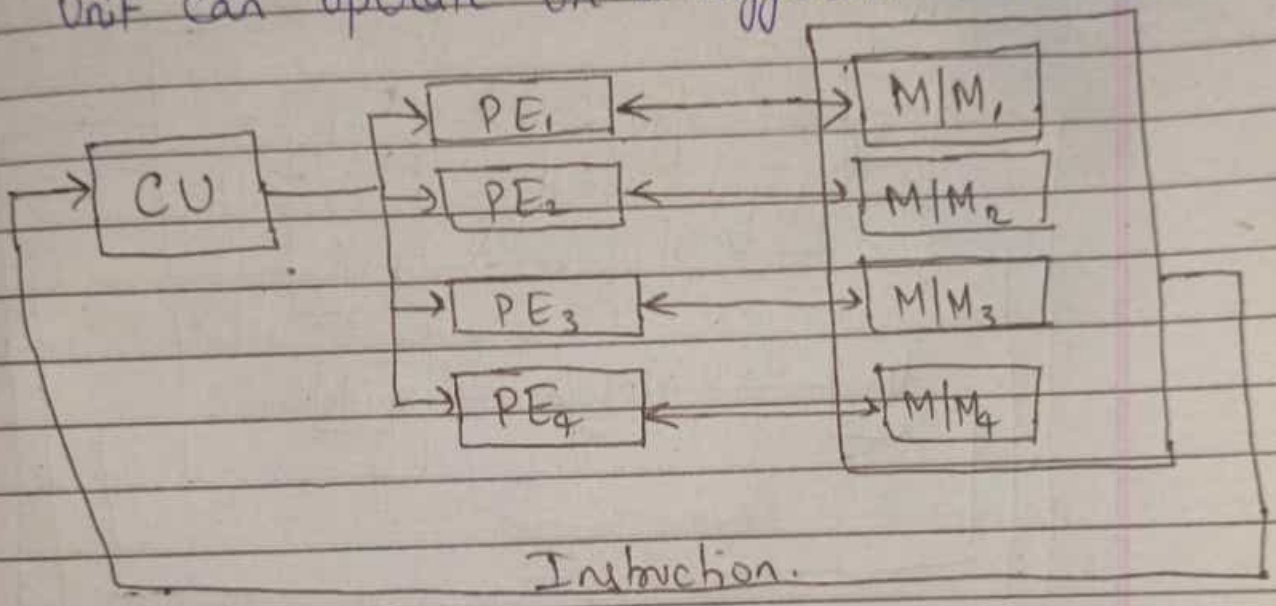


(a) SISD :- They are also called scalar processor i.e. one instruction at a time and each instruction have only one set of operands.



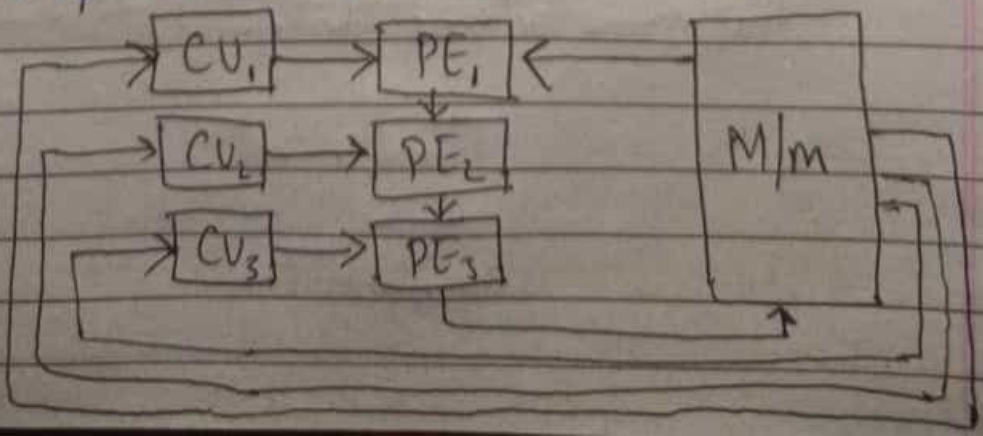
SISD have one control unit, one processor unit and single memory unit. In this instructions are executed sequentially.

(b) SIMD :- A type of parallel computer. Single Instruction is executed by different processing unit on different set of data. In this, all processing units execute the same instruction issued by control unit at any given clock cycle, also each processing unit can operate on a different data element.



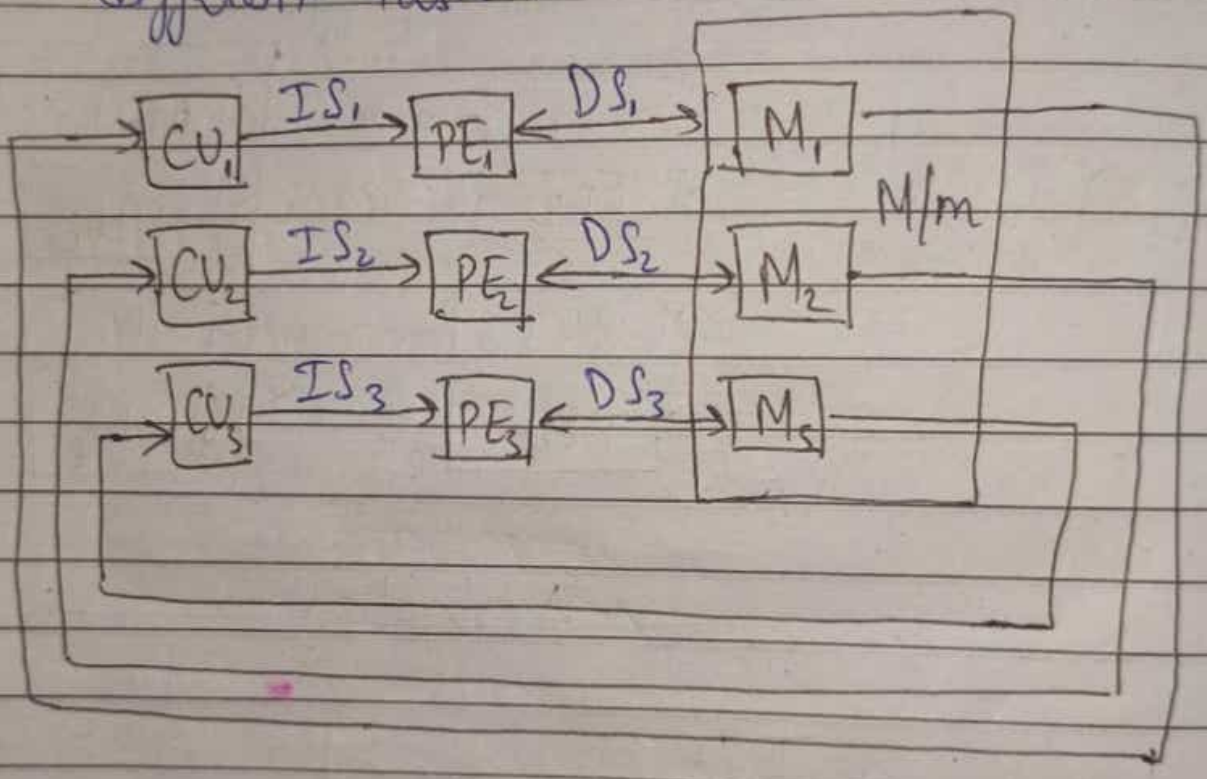
$I_s = 1 \quad D_s > 1$

(c) MISD :- A single data steam is fed into multiple processor unit. Each processing unit operates on data independently via independent instruction.



(d) MIMD :- In this, every processor may be executing a different instruction steam and different data steam too.

Execution can be synchronous or asynchronous. Different processor are there with each of them processing different task.

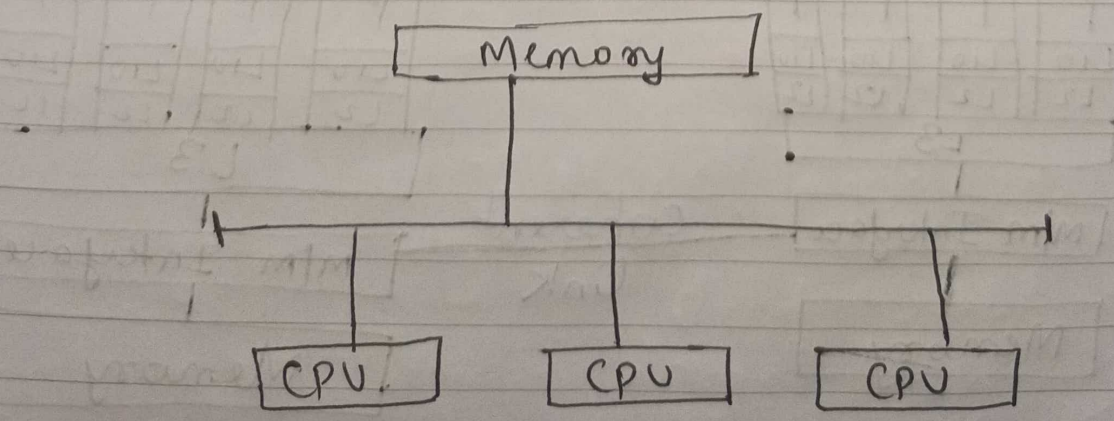


Shared Memory Computers :-

A shared memory parallel computer is a system in which a no. of CPUs work on a common, shared physical address space. Although transparent to the programmer as far as functionality is concerned, there are two variety of shared memory system that have very different performance characteristics in terms of main memory access :-

(i) UMA :- Uniform Memory Access system exhibits a "flat" memory model. Latency and bandwidth are the same for all processor and all memory location.
In this physical memory is uniformly shared by all the processors. All the processors have equal access time to all memory words So it is called UMA.

UMA 2 Categories :-
(a) Symmetric → equal access to all processors
(b) asymmetric → not equal access to all processor.



ii) ccNUMA :- On cache-coherent Non-Uniform Memory Access machines, memory is physically distributed but logically shared.
It is a special type of multiprocessor system that has mountable processor capabilities.
Mounting is a process by which a computer operating system makes a files and directories on a storage devices (e.g. CDROM, Harddisk, etc).

In ccNUMA, a Local Domain (LD) is a set of processor cores together with locally connected memory. This memory can be accessed in the most efficient way (i.e without resorting to a network of any kind). Multiple LDs are linked via a coherent interconnect, which allows transparent access from any processor to any other processor's memory.

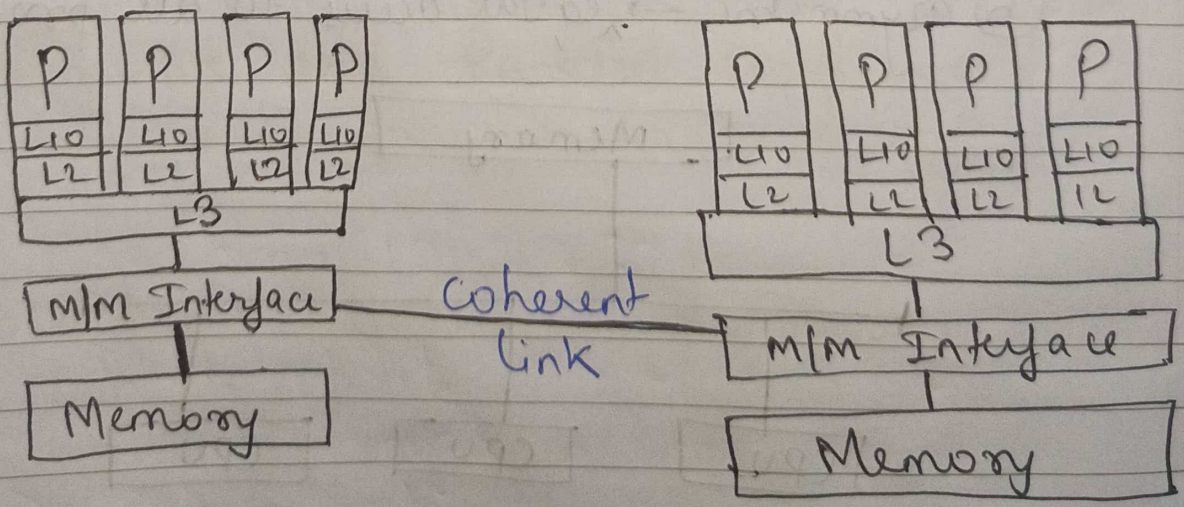
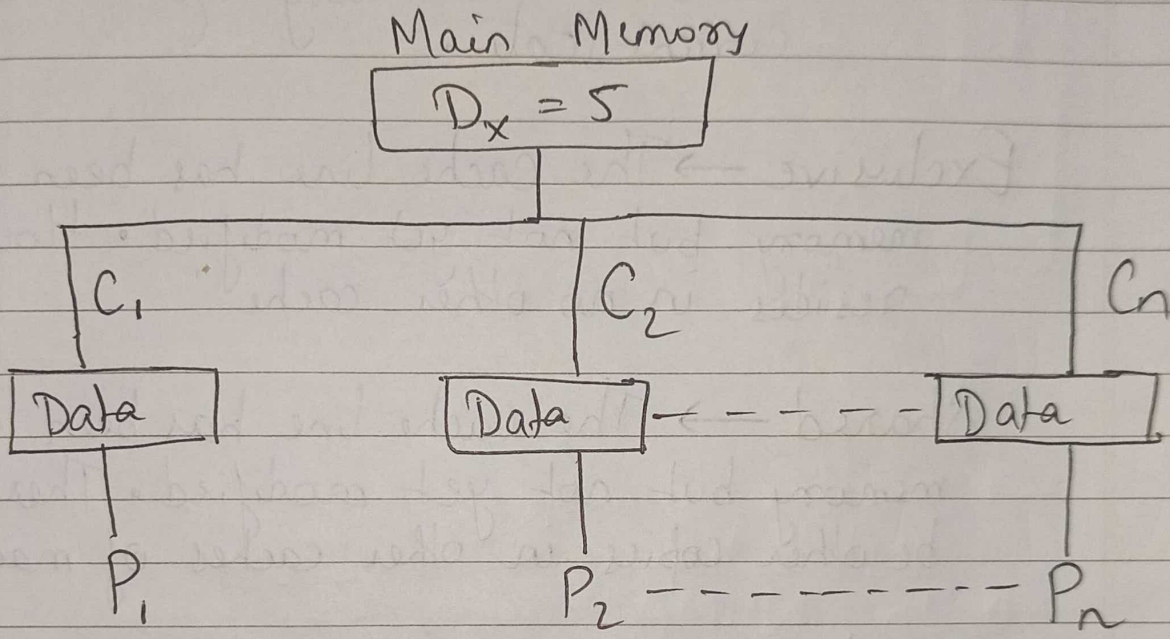


fig. A ccNVMA System.

Cache Coherence :-

Cache Coherence is the discipline that ensures that changes in the values of shared operands are propagated throughout the system in a timely fashion.



Cache Coherence occurs because copies of data might go in various cache lines and update in one may hamper the real data.

Solution :-

1) Write update - Write Through
2) Write update - Write back
3) Write invalidate - Write Through
4) Write invalidate - Write back.

→ Write update → update in $C_1, C_2, C_3 \dots C_n$ simultaneously.
→ Write through → simultaneously update in main m/m.
→ Write back → Jab tak Job khatam nhi ho tb tak update na kare.
→ Write Invalidate → Jaise hi C_1 me modify huga, waise hi sbko inform karega ki use mat karo.

To overcome Cache Coherence, we use MESI Protocol :-

Modified → The cache line has been modified in this cache and it resides in no other cache than this one. Only upon evichion will memory reflect the most current state.

Exclusive → The cache line has been read from memory but not yet modified. However, it resides in no other cache

Shared → The cache line has been read from memory but not yet modified. There may be other copies in other caches of machine.

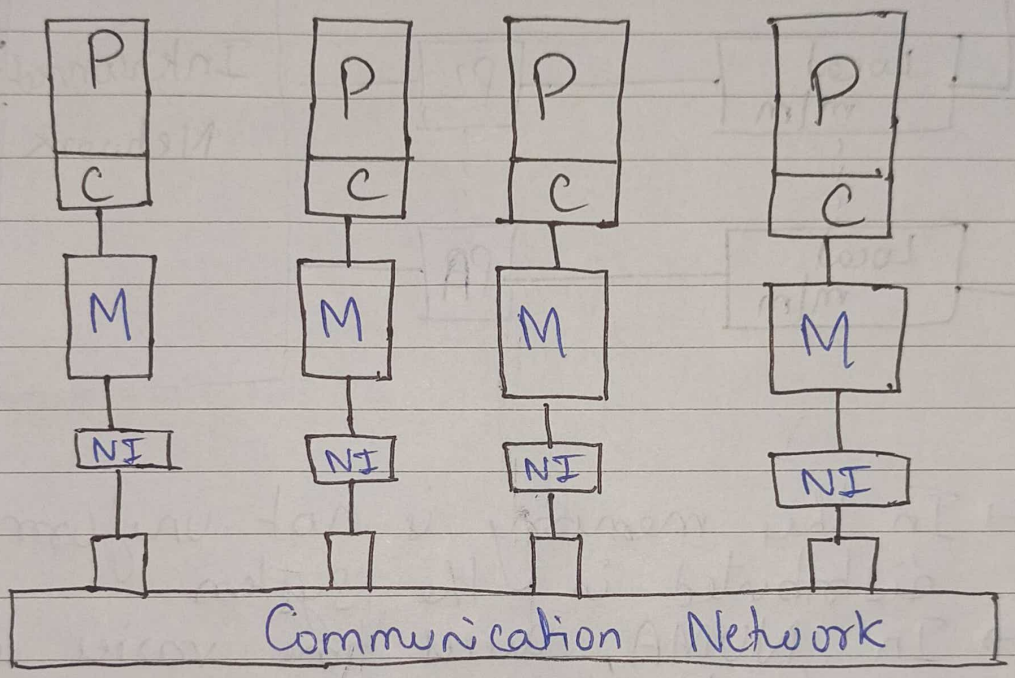
Invalid → The cache line does not reflect any sensible data. Under normal circumstances this happens if the cache line was in shared state and another processor has requested exclusive ownership.

Cache Write Policies :-

1) Write updat
2) Write through
3) Write back
4) Write Invalidate

Distributed Memory Computers :-

- In distributed memory Computers, each processors P is connected to exclusive local memory (i.e no other CPU has direct access to it).



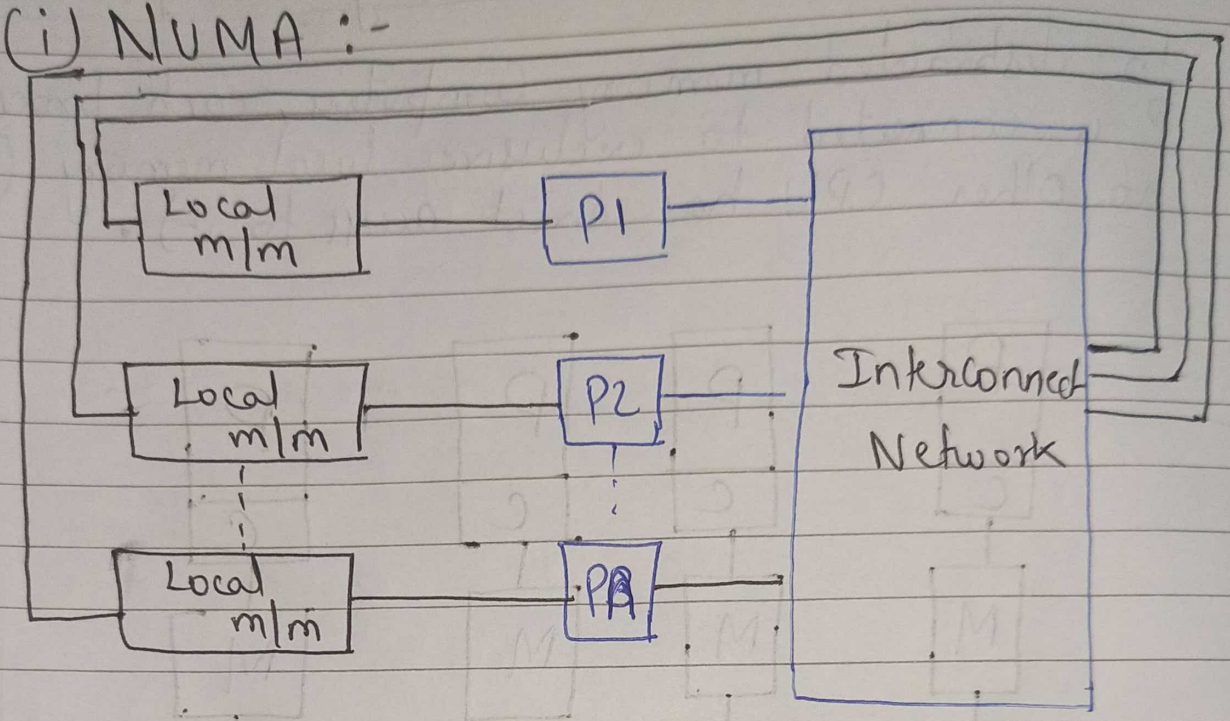
→ It is loosely coupled system because every processor has a own local memory.
It depend on (NO Remote Memory Access) NORMA concept.

Adv :- 1) Can use any of Ring, Star topology
2) Here, m/m is scalable with no. of process
3) ~~Info~~ Interference nhi h.
4) Cache coherence nhi hota.
5) Cost of effective

Disadv :- 1) More access time
2) Problem in Synchronization

Variety of Distributed Memory Computer :-

(i) NUMA :-



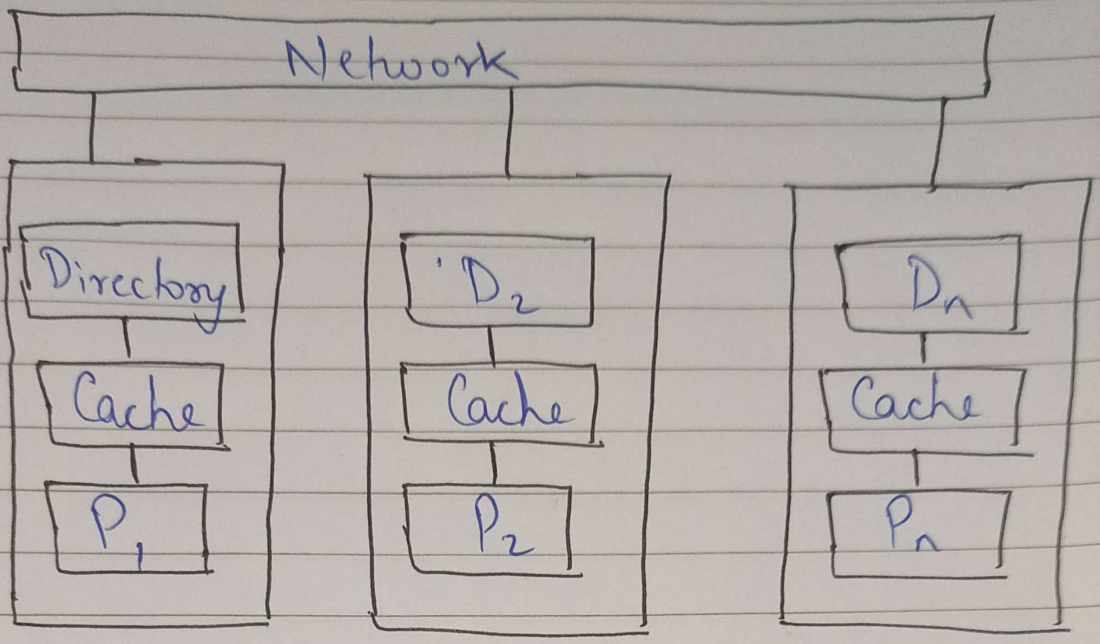
→ In this memory is not uniformly distributed in the system.

→ In NUMA, access time varies with the location of memory unit.

→ Access time is low when local memory is accessed by local processor.
But access time increase for accessing data stored in other processor's memory.

(ii) COMA (Cache only Memory Access) :-

A COMA machine includes several processing node connected by an interconnect network.



→ It is costly due to cache
→ It is a special case of NUMA model. In this too accs times varies with location of m/m unit.

→ Directory act as buffer b/w two Processor.

Hierarchical (Hybrid) Systems :-

Combination of both Shared & Distributed.

We can write about it on our own.

Adv, Disadv → take from distributed & Shared.

Diagram → Make combination of both.