Communication Cost in Parallel M/cs.

Cost of communication depends upon.

- model semantics.
- Network topology
- data handling & routing.

- Total time to transfer a message over n/w consists of following parameters.

— Start up time (t_s)

: time to spent for sending & receiving nodes. Executing routing protocols.

— Per hop time (t_h)

: time is a funct^n of hop & factors → no. of total routers.
n/w delay., switch latencies.

— Per word transfer time:

Overheads determined by the length of message.

Unit: 3

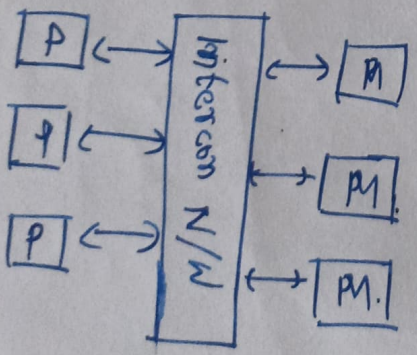
Principles of Message Passing Paradigm

- Introduction
  - The Basic Building blocks : Send / Rec. operation
- Send And Receive Protocols
- Message Passing Interface
  - min. set of MPI routines.
  - starting, Terminating the MPI library
  - Communicators — MPI_Comm, MPI_COMM_WORLD
  - Query Information
    - MPI_Comm_Size, MPI_Comm_Rank
      MPI_comm_size
- First MPI program.
- Sending & Receiving Messages.
- MPI datatypes.

→ Communication Model of Parallel Platforms ←

Shared data space

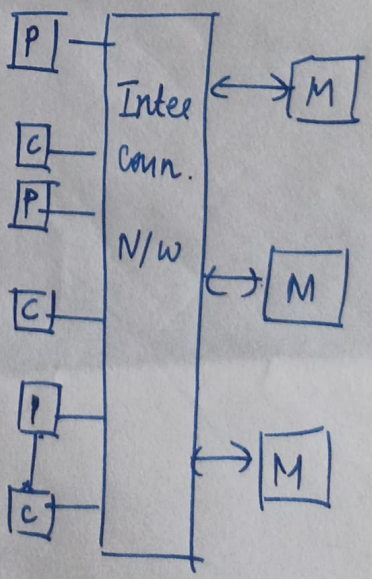
- Part of all memory is accessible to all processor
- time taken by a processor to access any memory word in the system global or local is identical the platform is the Uniform memory Access (UMA)
- else is Non uniform memory Access (NUMA)



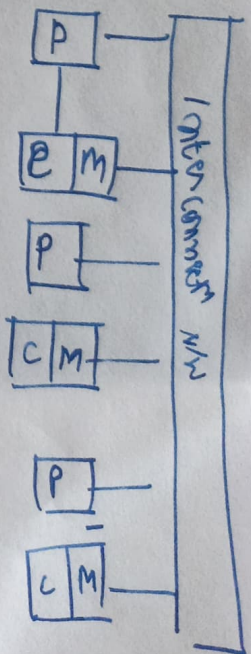
Shared adress space

Shared messages.

- All the processors share a common memory space and access the same data.
- Can be Implement using hardware & ∞ software.
- Ad: Exploit locality Reducing overhead latency.



UMA



NUMA

→ Message Passing

↳ communication method where processor has its own memory to send & receive message to and from processor.

↳ libraries: MPI (message passing interface)
PVM (Parallel Virtual M/c)

Ad: Scalabilities. to distribute across different distributed machines.

Libraries & function:

MPI send, MPI receive

→ Unit : 2 ←

Decomposition Techniques :

① Task & Dependency Graphs

Step 1 to implement parallel computing is to break complex problem into tasks.

Task : Indivisible unit of computation.

Representation : Directed Graph.

↳ It represents the control dependency

Task Dependency Graph

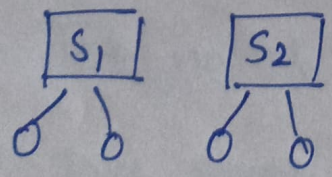
↳ Example: Data Query Processing.

→ Granularity ←

Fine ↙ ↘ Coarse.

① smaller comp.

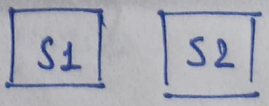
Ex: Op of comp. services.



② More objects holds less data.

① Large comp.

Ex: composite services.

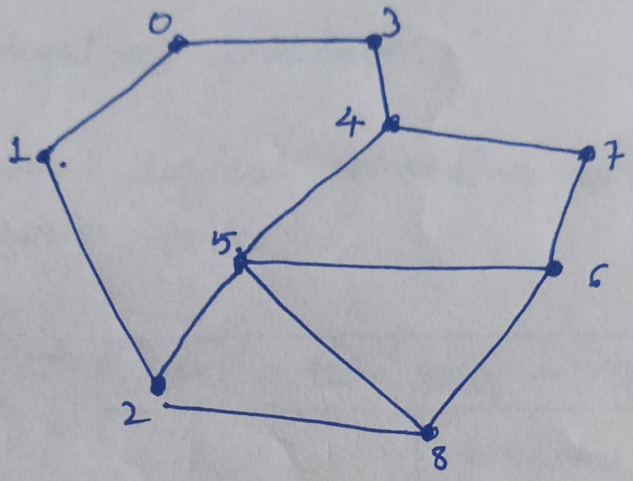


② Few object holds more data.

→ Task Interaction Graph ←

↳ Graph that captures the pattern of interactions among task.

↳ Represents data dependencies.



→ Process limitations of Parallel com ←

① Fine Granularity can not holds parallel time arbitarily small.

② Task Interaction graphs also a limiting factor.

→ Process & Mapping ←

→ The no. of task > No. of resources.

→ Thus parallel algo. must provide the mapping of task into processes.

Mapping is done from task to processes.

bcz: the typical prog. APIs do not allow the easy binding of task to physical processor.

↳ Task → Processes to map their process. to physical processor.

↳ Process : logical computing sense / agents that performs tasks.

→ [Task + task data + task code].

↓ Produces

[task of.]

↳ Processors : Physical hardware unit that perform task

→ Mapping is done by observing the task dependency graphs.

Mapping must minimize parallel time by :

→ Properly mapping independent task to dif process.

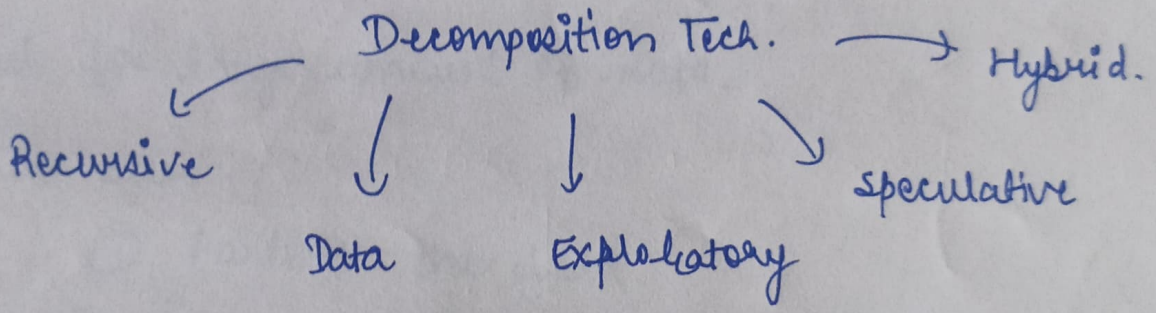
→ Assigning task on critical path to processes asap. they available.

→ Minimize interacth b/w known tasks.

Decomposition Techniques :

The procedure to break down the complex problem into simpler sub problems & task. which are indivisible.

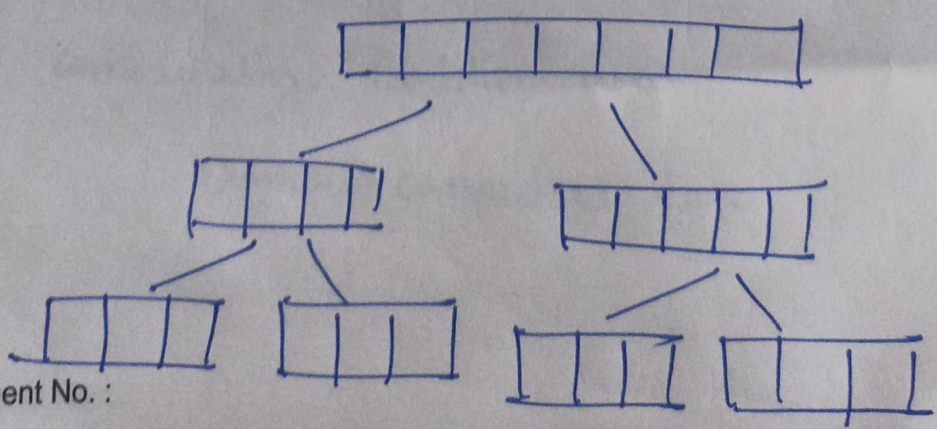
↳ There is no concrete path for solving all subproblem



① Recursive Decomposition

↳ Use to solve problems like divide & conquer.

↳ Complex problems are broken into sub problems. until a desired granularity reach.



```
min[A]=0 min = A[0];
for i:1 to (n-1) do
    if (A[i] < min) min := A[i];
endfor;
return min;
```

(2) Data Decomposition

↳ Used for large amount of data.

↳ Two steps
: (1) Partition the data.
(2) Induce comp. partition from data part.

↳ Type of Data to be partitioned.
↳ Input
↳ Output
↳ Intermediate. } Uses different data decom. techniques.

↳ computation Partition uses
: Owners computers Rule

Example: Matrix Multiplication

① Partitioning O/P data.

Applied : O/p data can be comp. independently as a function of o/ input.

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \longrightarrow \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

$C_{11} \longrightarrow A_{11} B_{11} + A_{12} B_{12}$

$C_{12} \longrightarrow A_{11} B_{12} + A_{12} A_{22}$

$C_{21} \longrightarrow A_{21} B_{21} + A_{22} B_{22}$

$C_{22} \longrightarrow A_{21} B_{12} + A_{21} B_{22}$

Conclusion

↳ Database of transacth is replicated across process each task can be independently acomp. w/o comm.

↳ Database of trans. is partitioned across process each task first computes partial counts.

↳ These task are then aggregated at app. task.

Input Data Partitioning

↳ This induces a task decom in which each task genrate partial count for all itemset.

↳ Combined subsequently for aggregate count.

Intermediate

Sequence of transact^n from i/p to o/p.

→ Owners Computation Rule ←

↳ Process assigned a particular data item is responsible for all its computation.

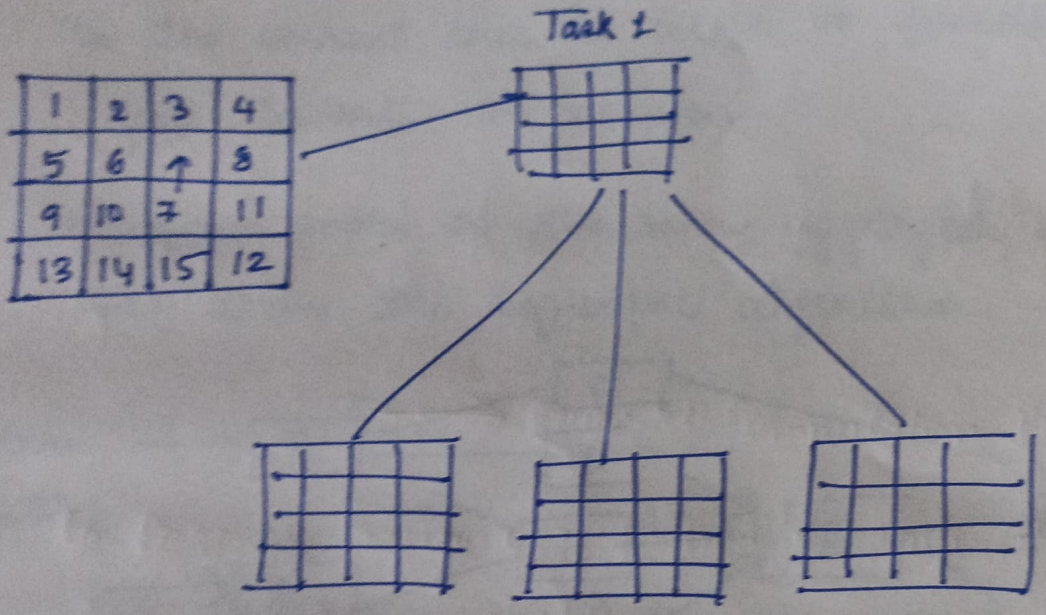
↳ Input Partitioning : Computations that uses i/p data are performed by process.

↳ O/p partitioning: computation that uses o/p are computed by the process

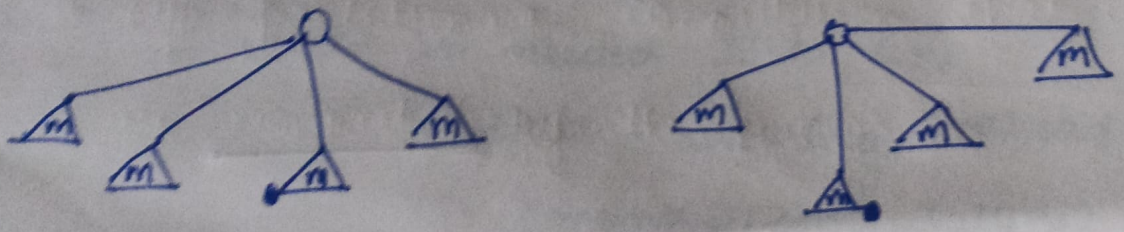
Exploratory Decomposition

- Used when computations that corresponds to search of space of solutions.

Ex: 15 - puzzle problem.



- Can cause change in amount of work done by the parallel formulation
- Change results in super or sub linear speedups.



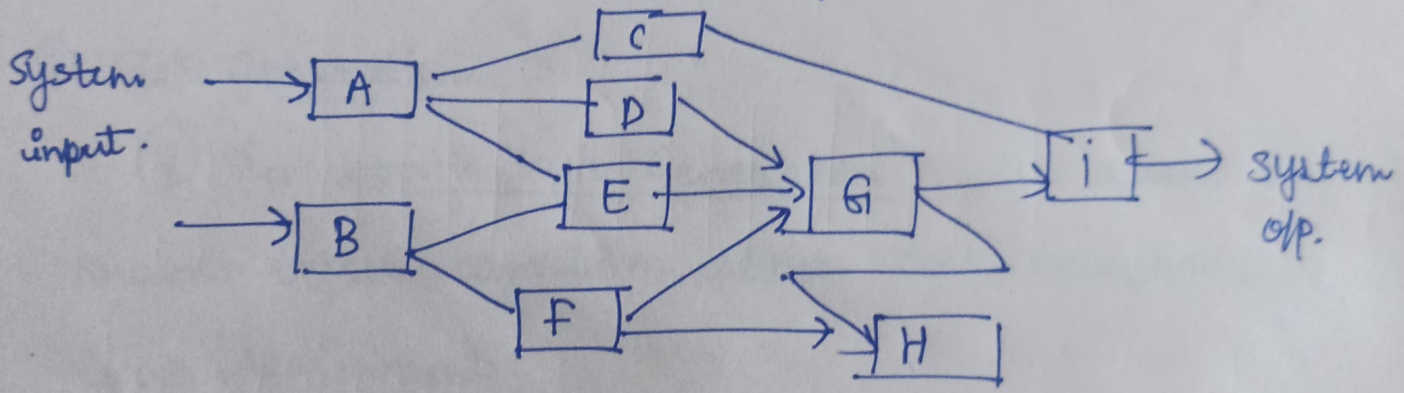
(a) (b)

→ Speculative Decomposition

↳ This decomposition is based on such problems in which solutions are based on the current task finishes.

↳ The current task is based to predict the next task towards the solution.

↳ Perform more or the same aggregate work (but not less) than the sequential algorithm



→ If predictions are wrong

↳ Work is wasted.

– Work is needed to be done to state - restoring overhead (memory / computations)

→ it may be the only way to extract concurrency.

The affect of good mapping on characteristics of Task

↳ The following are the four characteristics that can be have huge influence on the suitability of Task.

① Task Generation ② Task Sizes

③ Knowledge of Task size ④ Size of Data associated with task.

① Task Generation

① ↳ static task Generation : All tasks are known before algorithm starts task execution.

Ex: Assignments:

↳ Data Decomposition uses static task generatn.

Ex: Matrix Multiplication, LU factorizatn.

↳ Recursive Decomposition also uses static task generation (find min of a list of numbers)

② ↳ Dynamic Task Generation : All task are not available prior to before starting task execution algo.

↳ Recursive Algo. uses leads to dynamic task generat.

(quick sort : the task are generated dynamically

② Task size : Size of task it can be used to predict the the amount of time taken by a process
↳ complexity often depends upon task complexity uniformity.

③ Knowledge of Task size: If the size of task is known, this information can be used in mapping of task to processes.

④ Size of Data associated with a task: data associated with the task must be available to the process performing that task.

→ Characteristics of Inter Task Interactions ←

① Static V/s Dynamic
↳ The task generation would be static if: for each task the interactions happened at predefined time.
↳ and before algorithm execut^n the tasks are known priori
↳ Static T.P. can be programmed easily in Message Passi the reason is interaction in message passing require active involvement of sender & receiver.

• Dynamic :
↳ The task generation would be dynamic if: each task the interact^s happened at non-uniform time.
↳ Tasks are not explicity ~~non~~ kn not known priori before the execution of algorithm.
↳ Dynamic T.P. can not be easily programmed in Message Passing. because of inactive involvment of sender & reciver

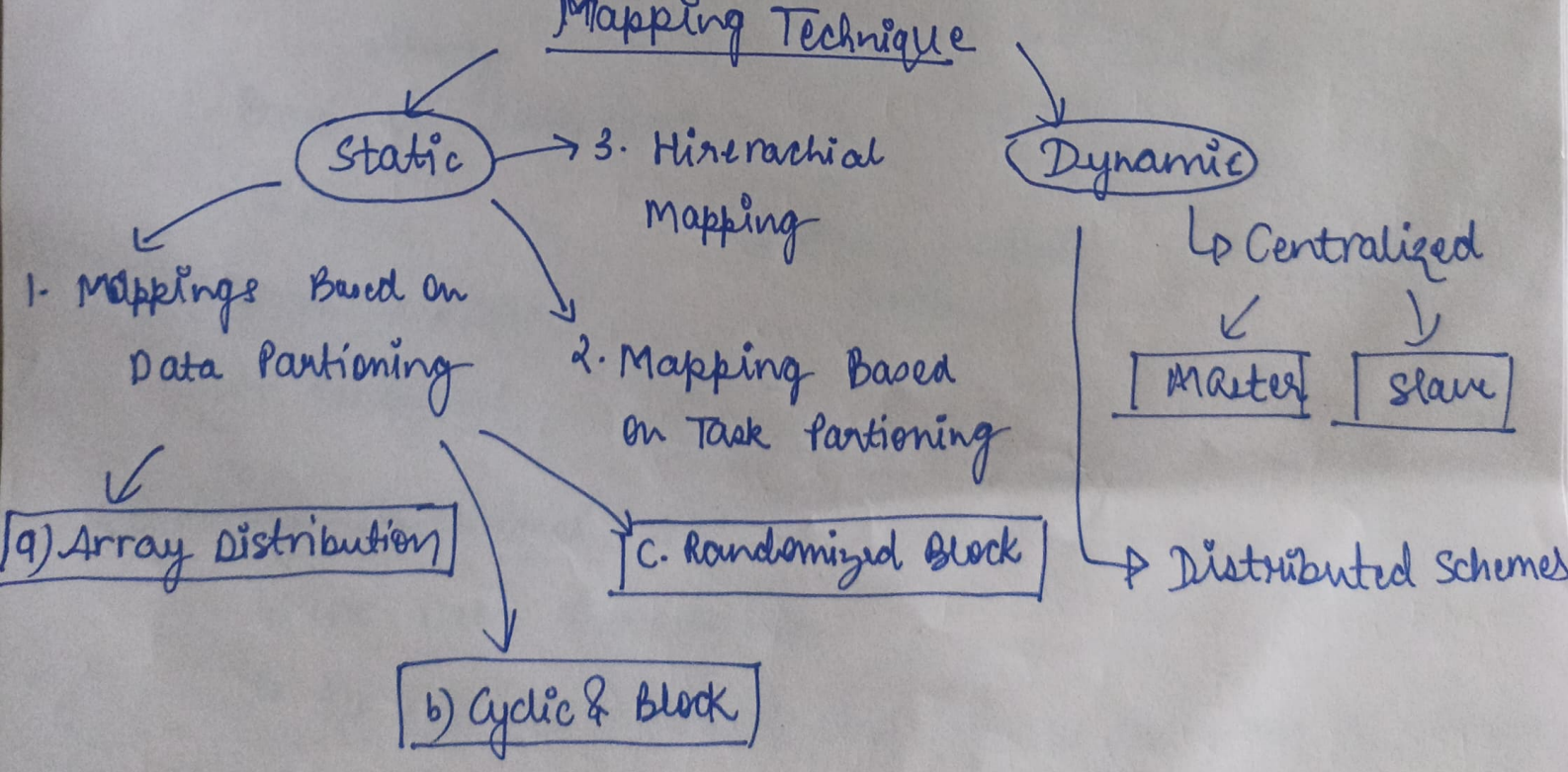
② Regular V/S Irregular

→ Way of classifying Interactions based on their spatial structure

Regular	Irregular
Patterns are regular	Patterns are irregular
Ex: Image Dithering	

⟶ Mapping Techniques for Load Balancing ←

① Mapping techniques is used to map from task to process so that the execution task is done within small elapsed time



Objective of Mapping

(1) Task should complete within shortest amount of time.

(2) Mapping must address following sources.

- ↳ load Imbalance.
- ↳ Intra process communication.

Static Mapping : Task are mapped prior to process.
Task generated statically.
Have uniform computational requirement.

Static Mapping

- ↙ Array
  1. Based on Data Partn
     - ↳ Array
     - ↳ by cyclic / Block
     - ↳ Randomized.
- ↓
  2. Based on Task Partition.
- ↘ Hierarical.

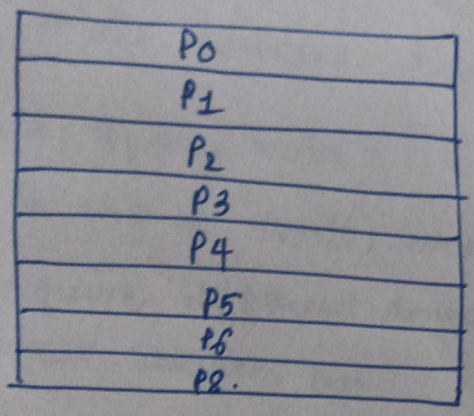
(1) Array Dis. Schemes

- ↳ use Data Decomposition
- ↳ I/P, O/P, Intermidiate data are in the form of arrays

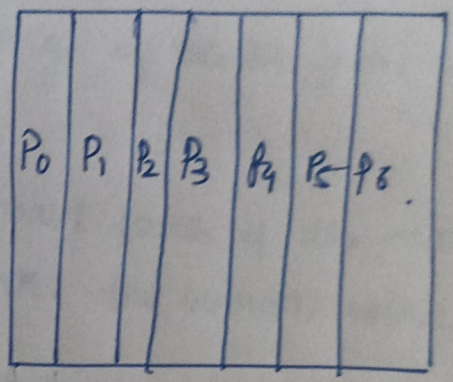
→ Based on Data Partioning

↳ combine partioning with Owners computation Rule.
↳ Simplest Data Decompostn : 1-D block distributn.

Row wise Distributn



Col. wise distributn



(9) Array Distribution

↳ User Data Decomposition : Owners Computation Rule.
↳ I/P ; O/P and intermediate data are in form of arrays.
↳ Can be generalized to higher dimensions.

[Array Block Distribution]

↳ Block is the simplest way to distribute an array
↳ This leads to have uniform contiguous portions.

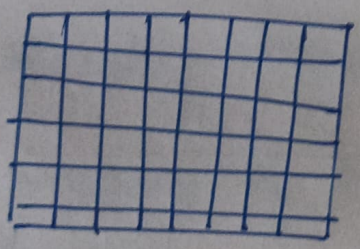
(b) Cyclic & Block . Ex: LU factorization

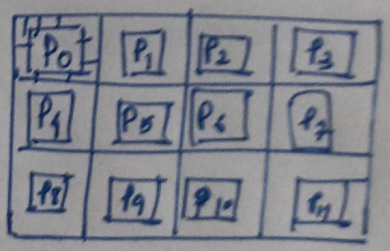
↳ Block-cyclic is variation of Block distribution. and can be used to alleviate the load balancing & idling problems.

↳ How to solve Idling problems. :

⇒ All process have sampling of tasks from all parts of the matrix.

↳ As a result, even if different parts of the matrix require different amount of work. the overall work on each process balances out.





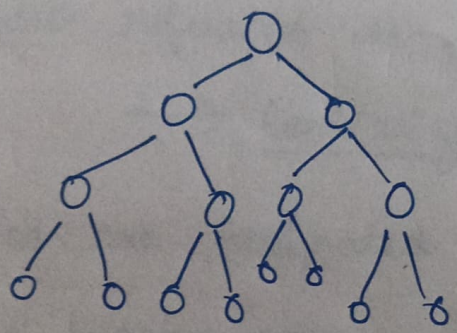
(c) Randomized Block.

↳ used where work has special patterns

↳ Block could contained non-zero elements due to which the other processes may not get work / or remained idle.

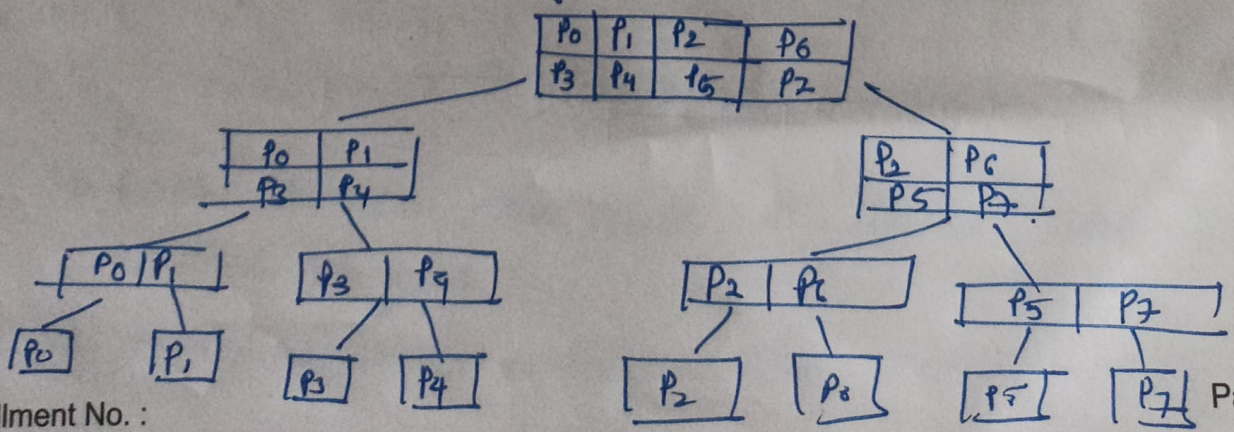
→ Here Based on Task Partⁿ ←

- Partitioning a given task dependency across processes.
- Determining optimal mapping for task dependency graph is NP-complete problem.



→ Hierarchical Mappings ←

- Single mapping have some limitations.
- task mapping of binary tree can not use a large number of process.
- For this reason task mapping is done at the top level and data partitioning with in each level.



Dynamic Mapping

- Centralized
  - Master
  - Slave
- Distributed.

↳ Dynamic Mapping do not know the task priori

↳ It is also referred as dynamic load balancing.

→ Centralized ←

↳ Processes are designated as 1) Master 2) Slaves.

↳ When a process runs out of work, it requests the master for more work.

↳ If the no. of process is increased.
then, master becomes bottleneck.

↳ To solve this process may pick up no. of tasks called chunks at a time.

↳ This called chunk scheduling.

↳ Selecting large chunks = load imbalancing.

→ Distributed ←

↳ Each process can send or receive work from another process.

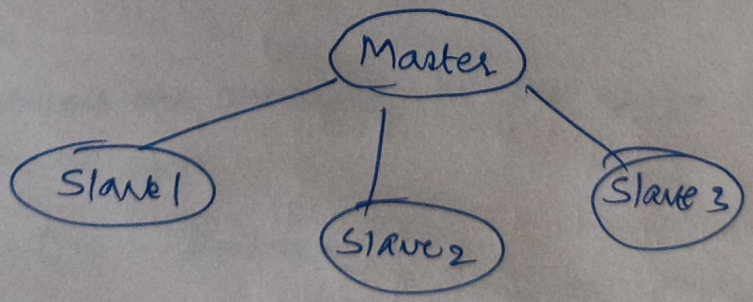
↳ This alleviates bottleneck in centralized system

→ Parallel Algorithm Models ←

→ It is a way of structuring a parallel alg. by
→ Selecting: Decomposition & Mapping Techniques.
→ Apply: Appropriate Strategy to minimize interact°s.

- Parallel Algo Models
  - 1. Master-Slave
  - 2. Pipeline / Producer-consumer
  - 3. Hybrid
  - 4. Work Pool.
  - 5. Data Parallel
  - 6. Task Graph.

(1) Master-slave Model



Master-Slave Model

↳ One processor is master : other units acts as slave.

Responsibilities of Master

1. Coordinate: activities among slave.
2. Generates: works to be performed.
3. Assign: work to slaves.
4. Synchronize: Activities of slaves after each phase.

↳ Model is suitable for shared add space & message passing paradigm.

Pipeline : consumer Model

↳ also called producer/consumer model.

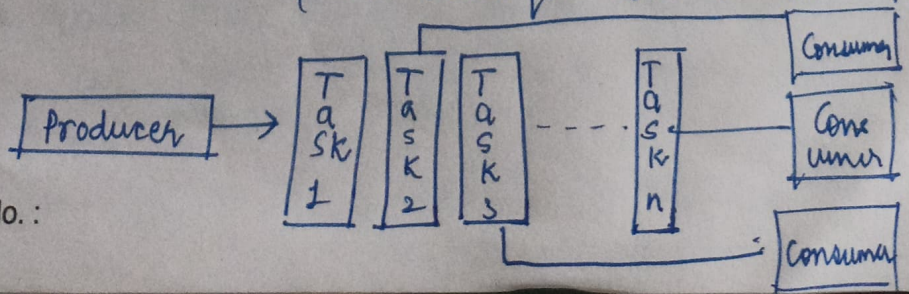
↳ Set of data passed through series of process. which perform some task.

↳ The process are arranged in the queue in the for Multi-D array tree, graphs or cycle.

O/P = Consumer.
I/P = Producer

} Same as concept of pipeline
O/P of 1st phase = I/P for next phase

↳ Interact^n Minimization Technique used is Overlapping Interact^n



Work Pool :

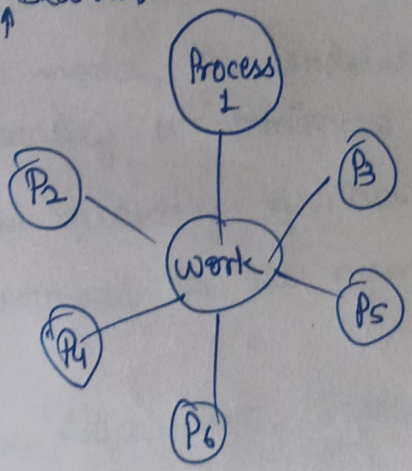
↳ Task are dynamically assigned to processes for balancing load.

↳ Therefore any process may potentially execute any task.

↳ This model is used when the quantity of data associated with the task is comparatively smaller than the computation associated with task.

↳ If the task is generated dynamically & decentralized assigning of task is done, the termination detection algorithm is required.

Ex: Parallel Tree Search



Data Parallel :

↳ Task are assigned to process.

↳ Each task performs similar type of operations on diff data

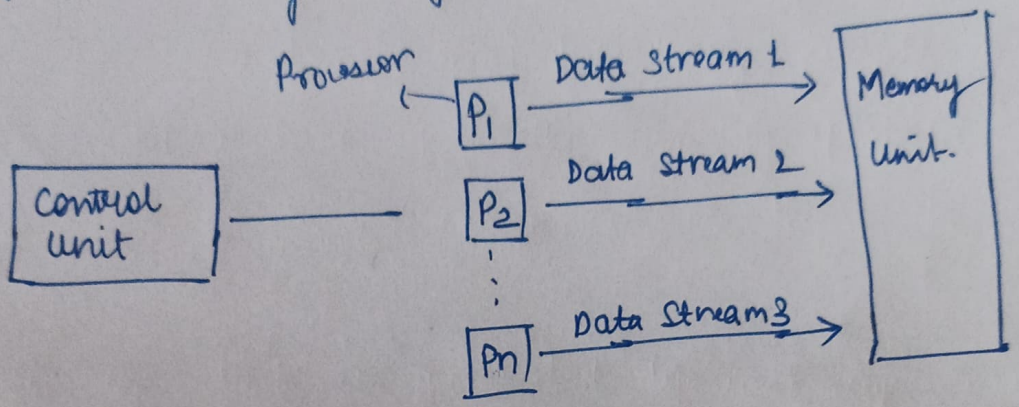
ie. SIMD (single inst. multi Data)

↳ Applied on: Shared Address Space, Message Passing

Shared Memory Paradigm

Characteristic : Intensity of Data parallelism increase with size of data.

use: for handling large data problems.



Task Graph :

- Parallelism is expressed by task graph.
- In this model, the correlation of task are utilized to promote locality or minimize interact^{n} cost.
- Use : the quantity of data associated with the task is huge compared to the number of computation associated with them.

Ex: Parallel Quick Sort, Sparse Matrix Multiplicat^{n}