High performance computing

Unit II

Q1 Explain granularity, con-currency & dependency graph.

→ Granularity:

1) The size of these tasks is expressed as the granularity of the parallelism.
2) The number and size of tasks into which a problem is decomposed determines the granularity of the decomposition.
3) The grain size of a parallel instruction is a measure of how much work each processor does, compared to an elementary instruction execution time.
4) In some applications a single operation is to be performed on many pieces of data. These operation are performed in parallel over the data set, generally with each Processing Element (PE) communicating with its neighboring PEs.
5) As per the defination of granularity, this task would be considered to have a small grandularity
6) With this context decomposing a computation into large number of small tasks is called fine-grained grandularity.

Concurrency:

1) concurrency releate to an application that is processing more than on task at

the same time.

2) Concurrency is an approach that is used for decreasing the response time of the system by using the single processing unit.

3] It is creates the illusion of parallelism however actually the chunks of a task aren't parallelly processed, but inside the application there are more than one task is being processed at a time.

4) It doesn't fully end one task before it begins ensuing.

5) Concurrency is achieved through the interleaving operation of processes on the central processing unit or in other words by the context switching

Dependency Graphs:

1) Is a directed acyclic graph, Typically a graph is a collection nodes & edges, the task-dependancy graph also contain nodes & edges

2) The node in task-dependancy graph is a task whereas edges between any two nodes represent dependancy between them.

e.g

There is edge exist between 2 nodes T1 & T2

If T2 must be executed after T1

Q List characteristics of tasks & interaction

→ a) Generation of task:

The task included in parallel algorithm are possible to generate either on prior basis termed static or dynamic

1) static task genration:

the task are defined before starting the execution the algorithm and specified order is to be followed by the algorithm in execution

e.g: matrix multiplication

2) Dynamic task generation:

The task to be perfomed are created dynamically based upon the decomposition of the data in certain situation

In these case, the task to be performed are not available before algorithm execution

e.g: recursive & exploratory algorithm

b) size of task:

1) Every task takes some amount of time for its completion.

2) Time duration required by the task to complete is termed as a size of task

3) size will be either uniform or non-uniform

4) Uniform size all tasks are of the same size. Matrix multiplication decomposition is of uniform size.

s) Decomposition of merge sort is example of non-uniform size.

c) knowledge of task sizes:
The task size knowledge is used for the choice of mapping. The task are mapped into process and prior knowledge of the task used in mapping.
e.g: where dynamic desision are required during processing

d) Data sizes:
The required data for performing a task should be made available when mappind it into processes
The overhead associated with the data movement can be reduced when the size of data as well as its memory location are available

Q Write down the decomposition techniques:

→ Data decomposition:

1) The large data sets involved in the problem can be divided into smaller part and process independent by several computers concurrently.

2) The datadcomposition is common techniquerused for concurrent processing of data.

3) There are basically 2 steps into their techniques. In 1st step, the overall i/p data is required for performing the defined computation is divided into multiple parts.

4) In 2nd step, it is based upon the division of overall computation into number of task handled on the partitione data.

5) The actual operation implemented an these task are similar but data upon which these operator work are different.

Example:-

Matrix multiplication:-

Consider the problem of multiplying two nxn matrix A & B to yield matrix C. The O/p matrix C can be partitioned into 4 task given below. Here each tasks computes one element of result matrix

Page No.
Date : / /

matrix A ⇒ $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$

matrix B ⇒ $\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$

matrix ⟶ $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$

matrix C ⟶ $\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$

$\begin{bmatrix} a_{11}\cdot b_{11} + a_{12}\cdot b_{21} & a_{11}\cdot b_{12} + a_{12}\cdot b_{22} \\ a_{21}\cdot b_{11} + a_{22}\cdot b_{21} & a_{21}\cdot b_{12} + a_{22}\cdot a_{22} \end{bmatrix}$

2] Recursive Decomposition:

1) As we know, divide & conquer are the strategy of solving the computational problem. It is based on 2 ideas: 1st approach to solve problem directly if it is small and if it is not solve then decomposee the problem into subproblem & solve the subproblems.

2) The recursive decomposition is based on providing concurrency in problem that can be handeled in divide & conqure strategy

3) The problem to be solved using recursion is divided into multiple subproblem, provided the each of

these sub problem in independent each sub problem can be solved individuall by dividing further into other sub problem using recursion.

In a programing a function or procedure calls it-self is called as recursion.

e.g

```
~~int~~ main()
  {
    int i;
    i=0;
    demo(i);
  }
demo(int count)
 {
   count--;
   printf("the value of the count is
%.d /n";count);
   if(count>0)
     demo(count);
   printf("the count is %.d \D";count);
}
```

3] Exploratory decomposition

i] sometimes the decomposition of the problem goes hand-in-hand with its execution

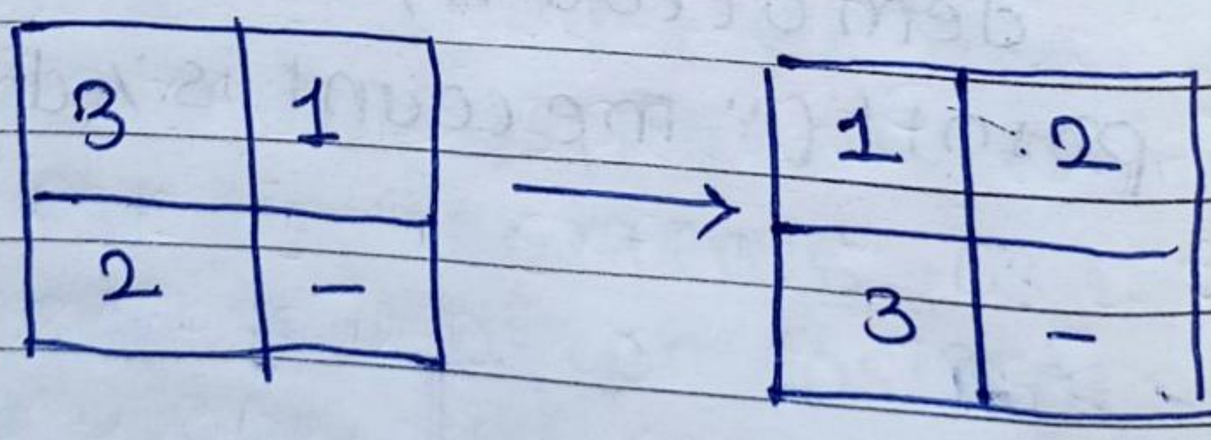
2] These problem typically involve the exploration search of a state space of solution.

3] In other words, your problem is running state & in the running state it is also decomposed which is referred to as exploratory decomposition.

4] for exampl : Puzzle problem

i] The puzzle problem is we have to transform from initial statito desired final srate.

ii] in this problem, there are 4 tiles numbered through 1 ro 4 are arranged in 2x2 grid shown



iii] one tile is left blank, so that moves can be made. The 2 possible moves here are up & left with blank tile. So we have divided initial problem into 2 sub problems.

iv] The same puzzle problem is possible with 4, 4×4, 8×8, 15×15 grid, only moves will be more.

v] So in exploratory decomposition, computation is split into tasks, each task searching a different portion of the search space.

4] Speculative Decomposition.

1] It is impossible to identify independent tasks whenever dependencies between tasks are not known in advance.

2] In this decomposition, the action to be taken is based on output of the preceding part.

3] This decomposition is used when a program may take one of many possible computationally significant branches depending on the output of other computation that precede it.

For example : Switch case

In switch case, among multiple available cases which case is to be considered is based on the input (expression) which has come from its preceding part.

compute expr:

```
switch(expr)
 {
    case1: compute t1;
    break;
    case2: compute t2;
    break;
    - - --
    - - -
 }
```

Here lots of option are available to choose, but which one to select is depened on previous preceding code part

4] One more example of speculative decomposition is topological sorting.

Q Explaing Mapping Techniques for load Balancing.

→ Mapping:

In parallel programing design it is required to specify where each task is to execute as we know process perform a task. In this approach of context used for task to process is called mapping

2] The mapping techniques is used to map tasks into processes so that parallel execution can be performed

3] The overall computation associated in problem to solved divided into number of task

4] These task are mapped out processes with the goal of completing execution in the minimum amount of time.

5] There are 2 techniques:

A] static mapping

In this technique, the mapping of task onto process is performed before execution of algorithm. This indicates that tasks are distributed among available processes prior to execution of algorithm.

Scheme for static ~~memory~~ mapping are as follow:

1) mapping Based on Data partitioning
2) Task graph partitioning
3) Hybrid strategies.

B] Dynamic mapping:

In this technique, the mapping of task onto the process is performed during the execution of the algorithm This indicate, the tasks are distributed among available processes when algorithm actually executes.

2) There are basic situation where dynamic mapping is applied. The 1st case is if tasks are generated dynamically then this mapping techniques is used.

3) Dynamic techniques is applied in situation where, large amount of data is associated with task. In this situation, dynamic mapping moves data among available processes.

4] There are 2 different type of classes dynamic mapping:

i] centralized dynamic mapping

In centralized dynamic mapping all executable tasks are maintained in a common central data structure they are maintained by a special process or a subset of processes.

ii] Distributed dynamic mapping:

In distributed dynamic mapping, tasks are distributed among processes which exchange tasks at run time to blance work.

Q explain any four method for containing interaction overheds.

→

1] The overhead that a parallel program incurs due to interaction among processes depends on many factors, such as, the volume of data exchange during interaction, the frequency of interaction, the spatial & temporal pattern of interaction etc.

2] General techniques used to reduce the interaction overhead are.

1] Maximizing Data locality:

a) The different processes required access to common input data or may be processes require data generated by other processes. In this case it will increase interaction overheads.

b) The interaction overhead can be reduced by using techniques that promote the use of local data or data that have been recently fetched.

various schemes of data locality:

1) minimize the volume of non-local data that are accessed

2) maximize the reuse of recently accessed data

3) minimize the frequency of accesses.

2] Minimizing contention & Hot-spots.

1) The data access and inter-task interaction pattern can often lead to contention that can increase the overall interaction overhead.

2) A frequent source of contention for shared data structures or communication channel is because of centralized schemes for dynamic mapping.

3) contention occurs when multiple task try to access the same resources concurrently.

3] Overlapping computation with Interaction

1] After an interaction has been initiated, the amount of time the processes spend waiting for shared data to arrive or to receive additional work can be reduced by doing some useful computations

2] There are many number of techniques that can be used for overlapping computation with interaction

3] The simplest technique is initiating early enough so that it is completed before it is needed for computation.

4] Replicating Data or Computation

1] techniques that are used to reduce interaction overheads

2] Replication of data:
multiple process may require frequent read-only access to shared data structure such as hash-table.
Replicate a copy of the shared data structure on each process

3] Replicating computation:
The processes in a parallel program often share intermediate results in some situations it may be more cost effective for a process to compute these intermediate result than to get them from another process that generates them.

Q ~~Write~~ Explain parallel Algorithm models in details.

→ Parallel Algorithm model:
It is used to describe the strategy for partitioning data how there data or processes
A model is used to provide appropriate structuring of parallel algorithm on the basis of 2 techniques.
1) selection of partitioning ← mapping techniques
2) proper use of strategy for interact minimization

1) Data Parallel model
a) It is simple algorithm model
b) In this model, the task are statically mapped onto processes and each task perform similar operation.
c) It is result of identical operation being applied concurrently on different data items
d) The work may be done in phases.
e) It can be implemented in both shared-address-space & message passing paradigm.
d) Interaction overhead can be minimized.
e) It solves large problem effectively
d) for example
Dense matrix multiplication.

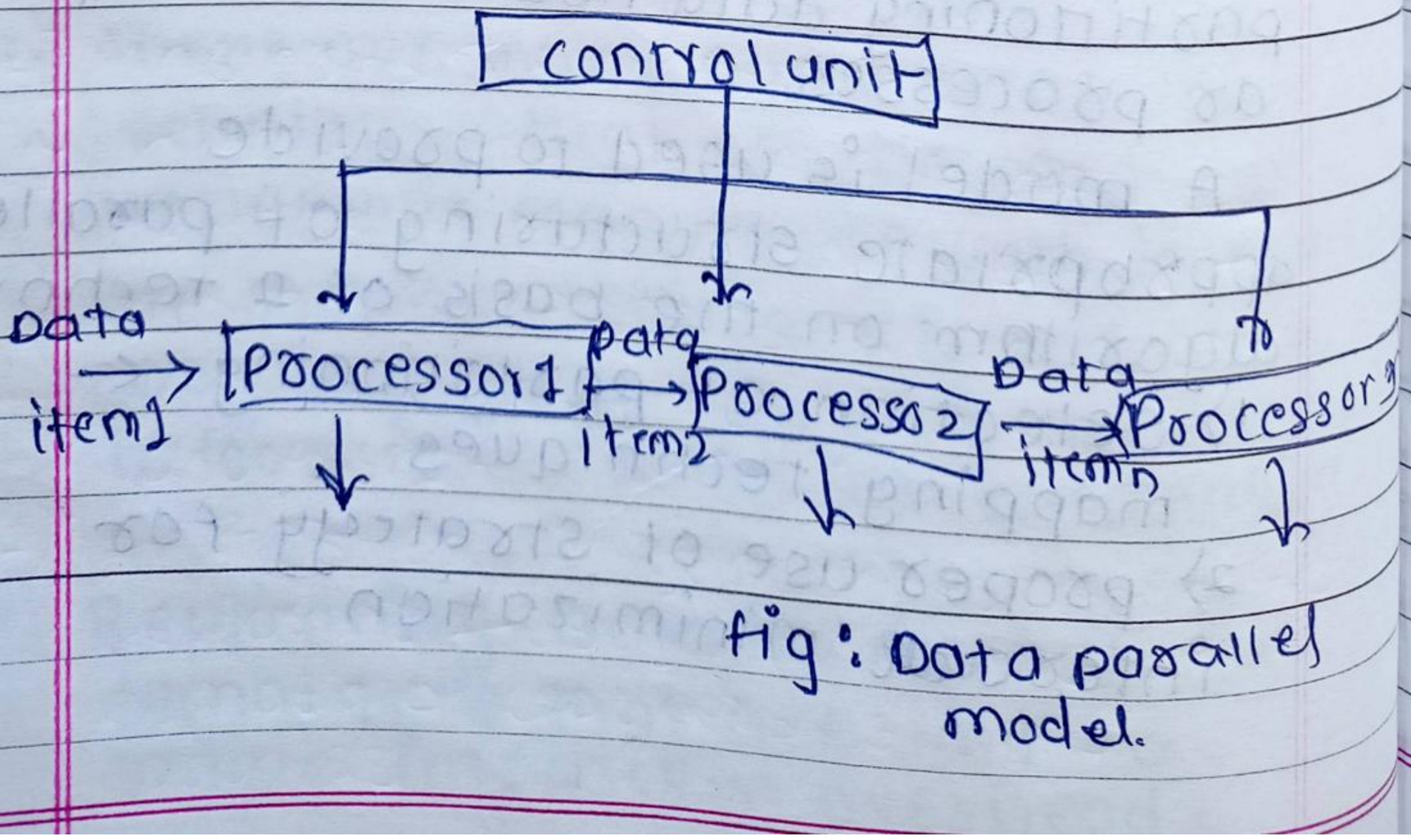


fig: Data parallel model.

2) Task Graph model:

1) starting from task dependency graph, the interrelationship among tasks are utilized to promote locality or to reduce interaction cost.

2) It is used to solve problem in which amount of data associated with tasks is large.

3) Tasks are mapped statiscally to optimize the cost of data.

4) For example: Parallel quicksort, sparse matrix factorization.

5) This type of parallelism that is naturally expressed by independent tasks is task dependancy graph called task parallelism

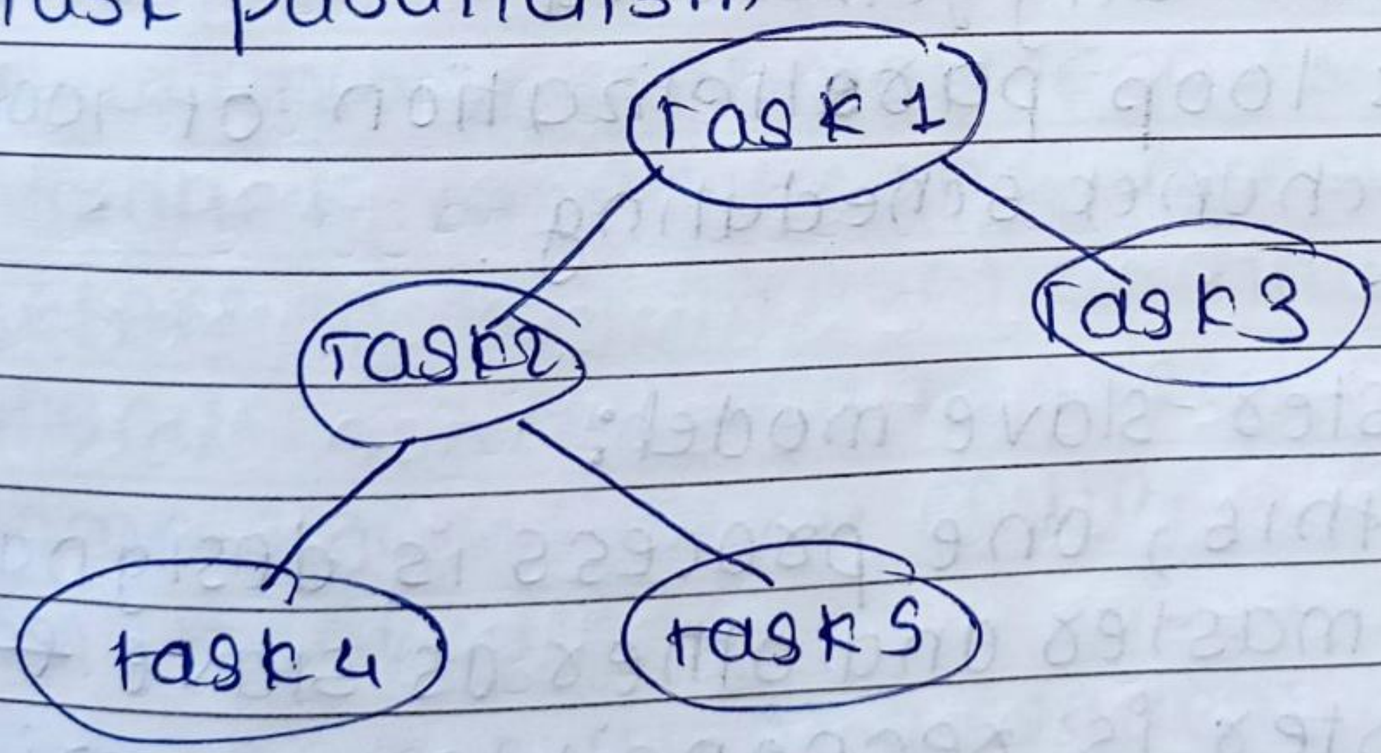


Fig: Task graph model.

2) Task Graph model:

1) starting from task dependency graph, the interrelationship among tasks are utilized to promote locality or to reduce interaction cost.

2) It is used to solve problem in which amount of data associated with tasks is large.

3) Tasks are mapped statiscally to optimize the cost of data.

4) For example: Parallel quicksort, sparse matrix factorization.

5) This type of parallelism that is naturally expressed by independent tasks is task dependancy graph called task parallelism

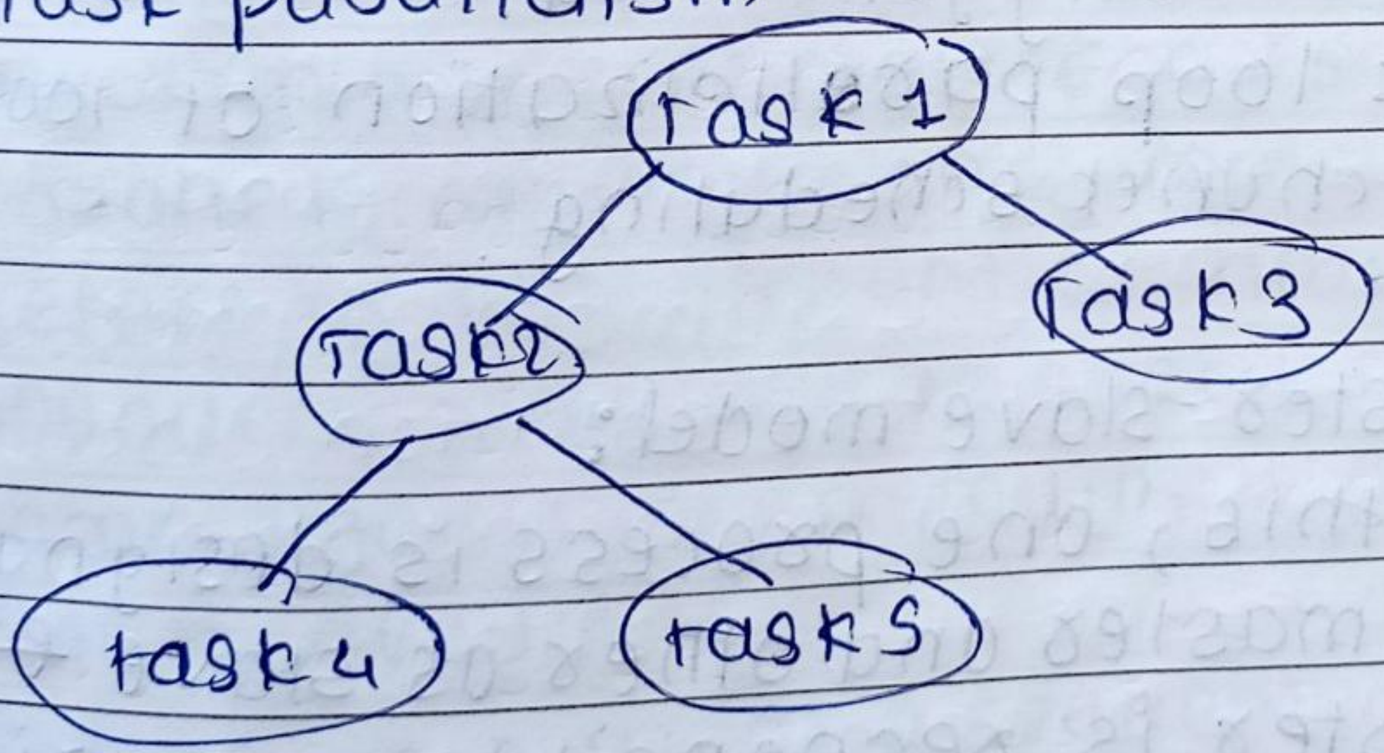


Fig: Task graph model.

a) The work pool model.

1) It is also called a task pool model

2) It is characterized by dynamic mapping of tasks onto process for local balancing in which task may be performed by any process.

3) No desired premapping

4) mapping may be centralized or decentrallized

5) pointer to task may be stored in physically shared list, queue, hash table or tree.

6) work may be statically available in the beginning or could be dynamically generated.

7) eg: loop parallelization of loops by chunck scheduling.

4] Master-slave model:

1) In this, one process is designated as master and other as slave & master is responsible to cooridinate the activites among all slaves processes.

2) In master, it generates work to performe & asign it to the number of workers / slaves process.

Page No.
Date : / /

3] In this, if work allocated to slave, the it complet by slave & slave is allocated to other job. It perform differen task.

4) master is responsible to synchron-ize the activities of the slaves after each phase.

5) The monster-salve model is generally suitable for shared address space and message passing paradigm.

Q Write short note on sequential & parallel computational complexity.

→

1) sequential computational complexit refers to the amount of time and resources required to execute ~~to~~ an algorithm on a single processor, while parallel computational complexity refers to the amount of time and resources required to execute the same algorithm on multiple processors simultaneouls.

2] The complexity of a sequential algorithm is typicall measured in terms of its tim complexity

3) The complexity of parallel algorithm is typically measured in term of its parallel speedup.

4) The parallel speedup is influenced by factor such as the number of processor available, the communication overhead between processors & the granularity of the algorithm.

5) the choice between sequential and parallel algorithms depends on factor such as the size & complexity of the problem being solved, the resources available for computation & the trade-offs between time & space complexity.

6) speedup of an algorithm is defined as the ratio of the worst-case execution time of the fastest sequential algorithm for a given problem to the worst case execution time of parallel algorithm.

7) The cost of the solution directly proportional to the number of processor an algorithm uses to solve a problem. ← Number of process used

$$P(n) = O(n)$$

8) Total cost : the sum of a parallel algorithm's time complexity & processor count is known as the total cost of that algorithm.

g] work-efficiency

the work, assuming that the cost of parallel machine is proportional to the numbe of processor in the machine

$T_{sequential} \leq T_{parallel} \cdot N_{processor}$.

q write short note on Anomalies in parallel Algorithm.

→

1) Anomalies in parallel algorithm refer to unexcepted or usual behaviod that can occur parallel algorithms

2) some common type of anomalies in parallel algorithms include.

a) Deadlock

A deadlock occurs when two or more processes are blocked, waiting for each other to release a resource or complete a task.

b) Race condition: A race condition occurs when the outcome of an operation depends on the timing or order in which processes are executed

c) Starvation: Starvation occurs when a process is unable to access a shared resource because other processes are monopolizing it.

d) load imbalance: Load imbalance occurs when some processes in a parallel algorithm are heavily loaded withe other are idle.

e) communication overhead: communicati overhed refer to the time & resources required to exchange data between parallel processes which can be significant in some algorithms.

3] To avoid these anomalies, it is important to carefully design parallel algorithm.

4) Additionally, implementing appropriate error handling mechanisms can help mitigate the impact of hardware failure during the execution of parallel algorithm.