UNIT-2

* Reference Model :-

Reference model focus on timing properties and resource requirements. It is classified into three characteristics -

1) Resource Graph : It describes the system resources.
2) Task Graph : It describes the application supported by the system.
3) Scheduling & Resource Management : It describes an algorithm that how the application system use the resources at all times.

* Processor & Resources :-

A job is executed by the operating system on a processor and may depend on some resources.

A processor is an active component on which job is scheduled.
EX:- CPU, Disk, Database server.
Every job must have one or more processors in order to execute and make progress forward completion.

A resource is a passive entity upon which jobs may depend.
EX:- memory, sequence number, database locks.
A job may need some resources in addition to the processor in order to make progress.
ρ (rho) types of resources have one or more unit and only one job can use each unit at once, use it then release.

* Parameter of Real-Time Work Load :-

There are three parameter of real time work load -

1) Temporal Parameter
2) Functional Parameter
3) Resource Parameter

Each job is characterized by its temporal, functional and resource parameter.

1) Temporal Parameter :-

It describes timing constraints and behavior of a model. The components of the temporal parameter are fixed release time, jittered release time, execution time, relative deadline, absolute deadline etc.

Fixed release time is a prior known time and jittered release time means the range of release time. The release time is not exactly known but within a certain range $[r_i^-, r_i^+]$; the range is known as jitter.

Execution time is the amount of time required to complete ~~its~~ the execution of job; when it executes alone and has all the resources it needs. It ~~is~~ depends upon complexity of job and speed of processor on which it is scheduled.

2) Functional Parameter :-

Scheduling and resource access decisions are made by functional characteristics of jobs and this is done by the value of functional parameters.

(i) Preemption of Jobs :-

If one job is running and the other job comes and wants processor time for execution, then the first job is suspended and the second job will be executed, this is called Preemption.

(ii) Criticality of Jobs :-

If a job is a urgent job so that it should be completed soon and the job has to be given higher priority or weightage.

During an overload, it is not possible to schedule all jobs, so first schedule critical jobs so that more critical jobs can meet their deadlines.

(iii) optional Execution :-

In some task, their are few jobs are optional. During overload, we can discard optional jobs when it is not possible to execute all of the jobs in time. Then the system performance degrade but the function satisfactority.

(iv) Laxity Function :-

By laxity function, we can differentiate whether timing constraints are soft or hard. So we can schedule according to soft or hard deadline.

3) Resource Parameter :-

Basic components of the system are processors and resources. In addition to a processor a job may also require some resources. The resource parameter provides the information that is needed to support resource management decision like units of each resource type required by the job and the time interval when the units are required.

(i) Preemptivity of Resources :-

A resource parameter is preemptive, when a unit of a non preemptable resources is allocated to a job then the other job must wait until the job complete its use. The lock on a data object in database is an example of non preemptable resource.

(ii) Resource Graph :-

Resource graph describes the configuration of the resources. Edge in the resource graph represent the relationship among the resources. There are two types of edges in the resource graph -

a) Is-a-part of edge :- Parent child relationship

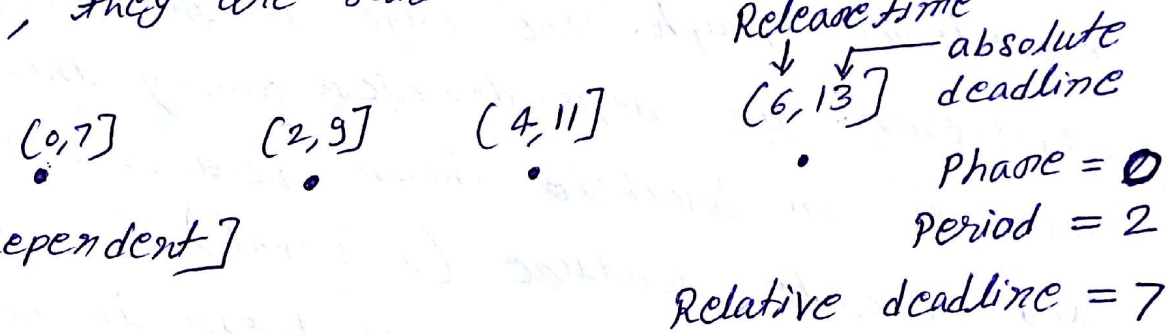
b) Accessibility Edge :- It shows that one component can access the data of another component.

* Periodic & Aperiodic Task Model :-

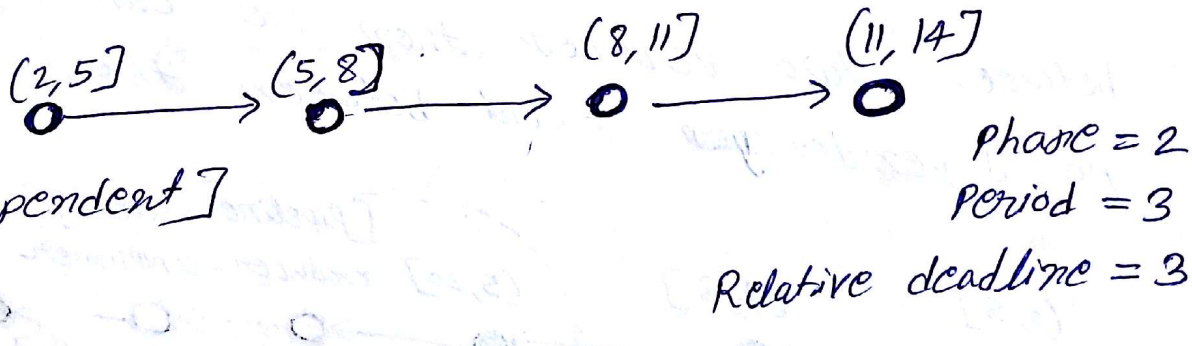
Refer UNIT-1 notes

* Precedence Constraints & dependencies :-

The jobs in a task, may be constrained to execute in a particular order, this is known as a precedence constraint. If the job can execute in any order, they are said to be independent



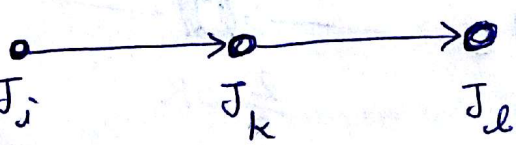
[Independent]



[dependent]

1) Precedence Graph & Task Graph :-

Precedence graph represent the precedence constraints among jobs using a directed graph. Each node represents a job and directed edge shows relations like from J_i to J_k, here J_i is an immediate predecessor of J_k.



It can also be defined on $(J, <)$.

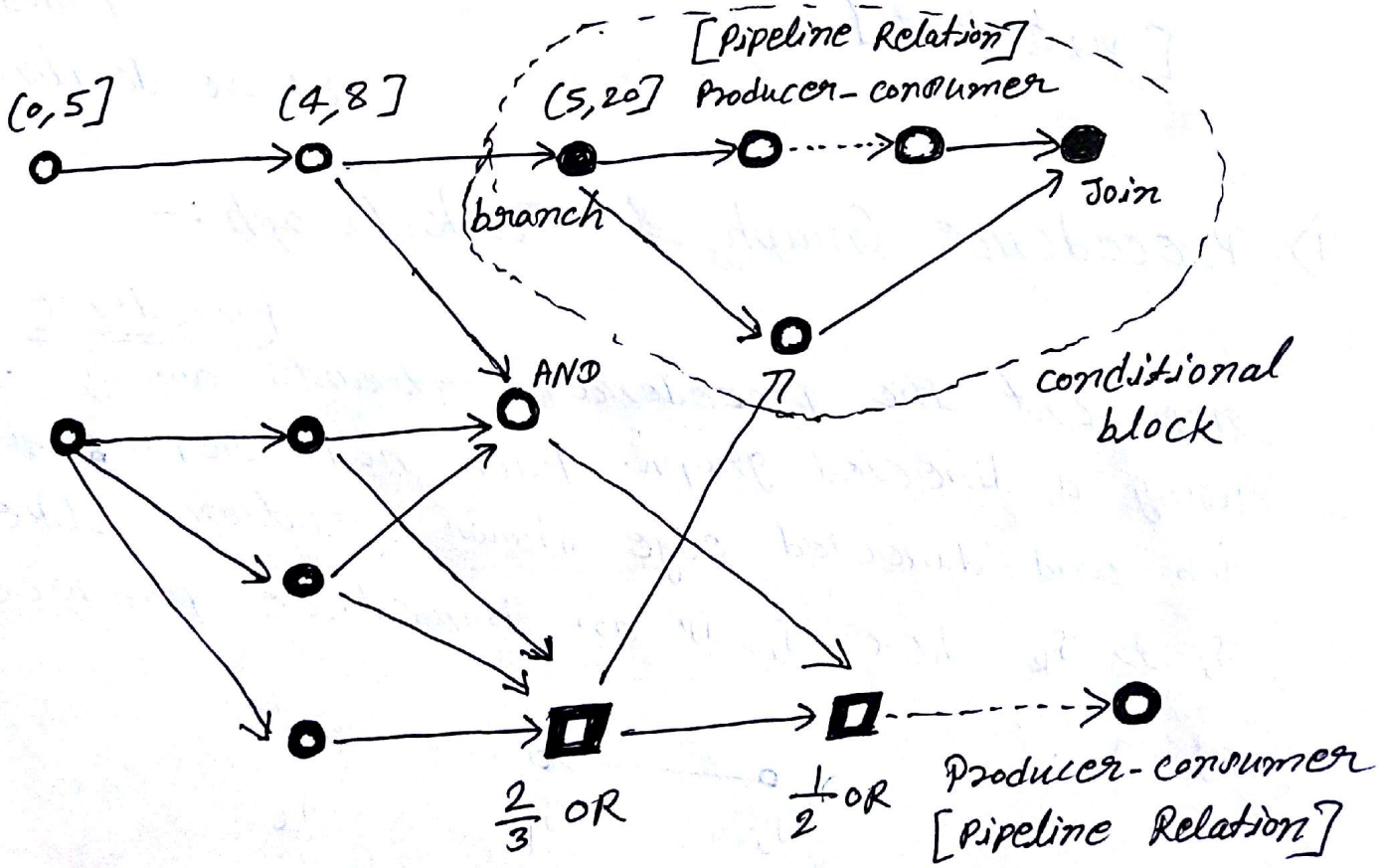
$$J_i < J_k < J_l$$

$\Rightarrow$ J_i is immediate predecessor of J_k and predecessor of J_l.

$\Rightarrow$ J_l is immediate successor of J_k and successor of J_i.

Task graph is an extended precedence graph. The edge in the task graph represents the dependencies among them and number in brackets shown release time and absolute deadline (interval).

If their is no edge between two vertices then we can say that no dependencies exist between them.



2) Data Dependency :-

Data dependency cannot be captured by a precedence graph. In a task graph, data dependency among jobs are represented by data dependency edge among them. There is a data dependency edge from a vertex J_i to J_k in the task graph, if the job J_k consumes data generated by J_i or the job J_i sends message to J_k.

(i) Temporal dependency :-

The difference between completion time of two jobs is called temporal distance. Jobs are said to have temporal constraints distance, if these temporal distance must be no more than some finite value.

EX:- The display of video frames with audio. when the video of that person speaking to have a lip synchronization, then time between display of each frame and generating corresponding audio segment must be no more than 160 ms.

(ii) AND/OR Precedence Constraint :-

A job with more than one immediate predecessor must wait untill all its immediate predecessor have been completed its execution, we call such jobs AND jobs and dependencies among them AND precedence constraints. AND jobs are represented by unfilled circle in the task graph.

In constraint, an OR job is one which can begin at or after its release time, provided ~~all~~ ~~or~~ ~~some~~ one or some of its immediate predecessor have been completed. OR jobs are represented by unfilled square. Here 2/3 labelled means execution begin as soon as two out of its three immediate predecessor complete its execution.

(iii) Conditional Branch :-

In a task graph, if edges expressing OR constraints, then ~~one~~ only one out of all the immediate successors of jobs is to be executed, is called branch job and ends at the vertex representing the join job. whole block is called conditional block. Branch jobs and join job are represented by filled circle.

(iv) Pipeline Relationship :-

dependency between a pair of producer consumer job, where consumer takes output of producer. In graph vertices are producer and consumer.

In the task graph, pipeline relationship between jobs are represented by dotted edge between jobs. There is an edge from J_i to J_k, if the output of J_i is piped into J_k.

* Real Time Scheduling :-

Real time scheduling algorithms are classified based on how the scheduling points are defined. The classification are :-

1) Clock Driven
2) Event Driven
3) Hybrid

A scheduling is an algorithm for ordering the execution of the task on a processor according to some predefine criteria.

Scheduling Criteria :-

In choosing which algorithm to use in a particular situation, we consider the following properties -

(i) CPU Utilization :-
we want to keep the CPU as busy as possible. In real time system range of CPU utilization is 40% to 90%.

(ii) Throughput :-
The number of processes that are completed per ~~time~~ unit time, is called throughput.

(iii) Turn Around Time :-
The interval from the time of release to completion time is turn around time. TAT = waiting time + Execution time

(iv) waiting time :-

waiting time is the interval between release time and responce time.

In non-real time system the typical goal of scheduling is to maximize average throughput and minimize average waiting time of task.

In real time scheduling, the goal is to meet the deadline of every task by ensuring that task can complete its execution by specific its deadline.

Scheduling Algorithm :-

1) Clock Driven :-

The clock driven schedulers are those in which the scheduling points are determined by the interrupts received from a clock.

(i) Table Driven
(ii) cyclic

This type of schedulers are also called offline or static schedulers; because these schedulers fix the schedule before the system starts to run. That is, the scheduler predetermines which task will run when.

Table driven schedulers usually pre compute which task would run when and store this schedule in a table at the time the system is designed.

Task	Start Time
T_1	0
T_2	3
T_3	10
T_4	12

Fig : Table driven

Task No.	Frame No.
T_3	F1
T_1	F2
T_3	F3
T_4	F2

Fig : Cyclic scheduler

Cyclic schedulers are being extensively used in the industry; where tasks are repeated periodically.

2) Event Driven :-

In event driven scheduling, the scheduling points are defined by task completion and task arrival events. This class of schedulers are normally preemptive, that is, a higher priority task when become ready, preempts any lower priority task that may be running.

(i) simple priority based
(ii) RMA (Rate Monotonic Analysis)
(iii) EDF (Earliest Deadline First)

3) Hybrid :-

In hybrid schedulers, the scheduling points are defined both through the clock interrupts and the event occurance.

Ex Round Robin

Round Robin is a preemptive scheduling method, here the ready tasks are held in a circular queue. once a task is taken, it run for certain fixed interval, called time slice. If a task does not complete

complete within its allocated time slice, then it is inserted back into the circular queue.

It is less proficient to schedule real time tasks; because it treats all tasks equally and all tasks are assigned identical time slices irrespective of their priority, criticality or deadlines. So task with short deadlines might fail to complete on time.

* Weighted Round Robin :-

In round robin approach every job gets equal time slice, but in weighted round robin, rather than giving all jobs equal share of processor, different jobs may be given different weight. A job with weight w get wt time slice in every round.

Suppose there are 3 jobs having weight –

$J_1 \Rightarrow w_1 = 2$

$J_2 \Rightarrow w_2 = 1$

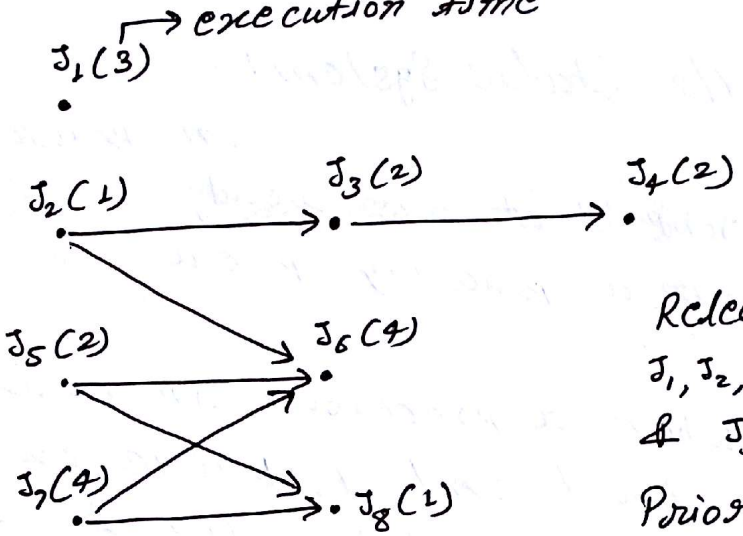
$J_3 \Rightarrow w_3 = 1$

& their time quantem or time slice is 1. So J_1 takes 2 slices while J_2 and J_3 gets one-one time slices and ~~length~~ length of round is equal to the sum of weights of all the ready jobs.

✳ Priority Driven Approach :-

In this approach, scheduling decisions are made using events such as release and completion of job. If two or more jobs are conflicting for schedule, then processor will allate to job, which have higher priority.

EX



Release Time

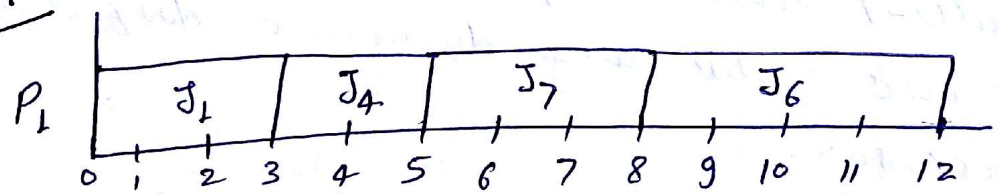
$J_1, J_2, J_3, J_4, J_6, J_7, J_8 = 0$

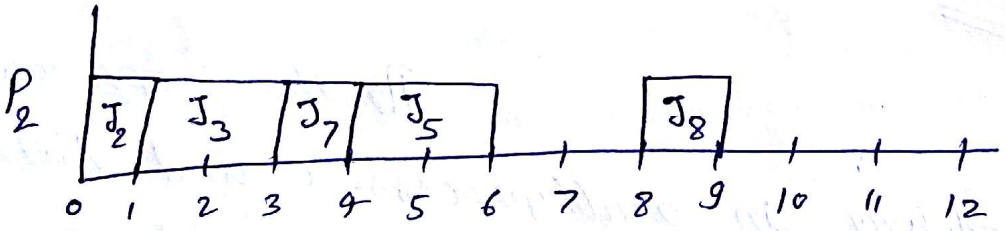
& $J_5 = 4$

Priority ⇒

$J_1 > J_2 > J_3 > \cdots > J_8$

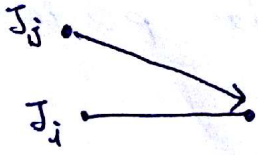
Preemptive





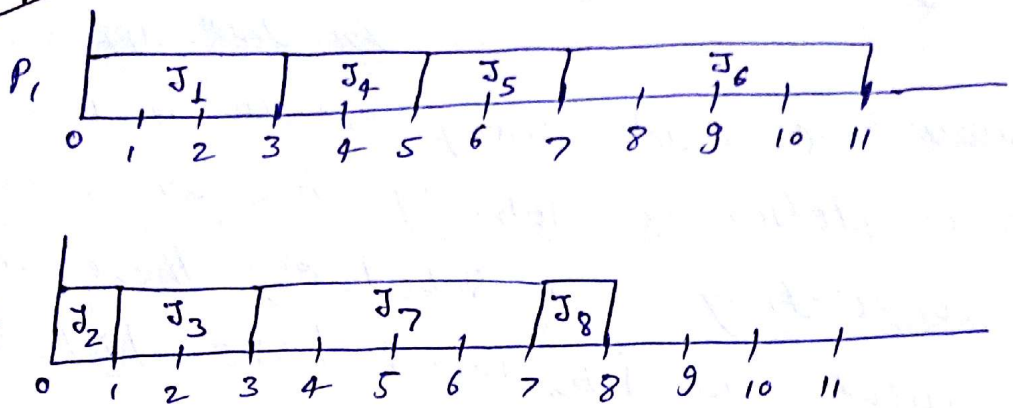
Here

$J_i \bullet \longrightarrow \bullet J_k$ ⇒ It shows J_k does not start its execution till J_i complete execution.



⇒ J_k start its execution after both J_i & J_j complete its execution.

Non-Preemptive



* Dynamic V/s Static System :-

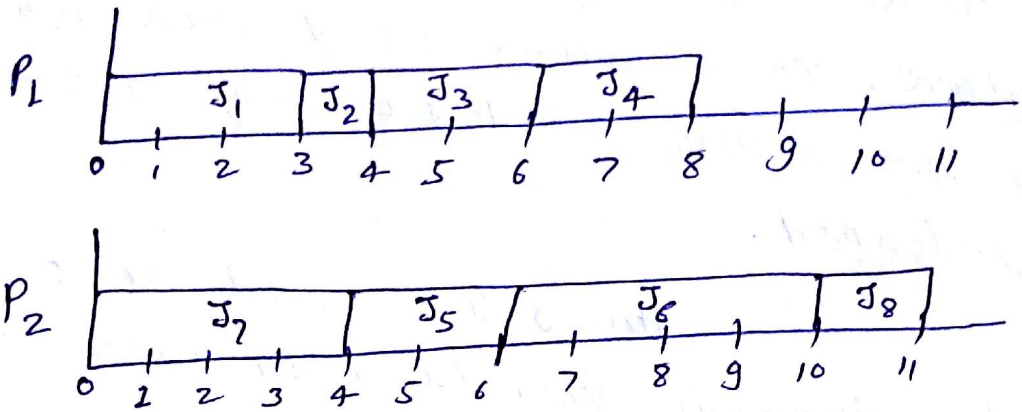
In priority driven approach jobs that are ready for execution are placed in a priority queue common to all processors.

When a processor is available the job at the front end of queue execute on the processor. We will refer to such a multi-processor system as a dynamic system. Because jobs are dynamic dispatch to the processor.

EX :- Previous example of Preemptive

Another approach to scheduling in multiprocessor and distributed system is to partition the job in the system into subsystem and bind the subsystem ~~into~~ statically to the processors. Such a system is called static system because it performs statically.

EX Bind J_1 to J_4 in processor P1 & bind J_5 to J_8 in processor P2.



* Offline V/s Online Scheduling :-

Offline scheduling is computed before the system begins to execute and the computation is based on the knowledge of the release time, processor time, resource requirements of all the jobs for all times.

This approach is inflexible, because it is possible only when the system is deterministic, that is the system provides set of function like release time processor time. Resource demands of all jobs are known and do not vary time to time.

EX :- Clock driven scheduling

In online scheduling, the scheduler makes each scheduling decision without knowledge about the jobs that will be release in the future. The parameter of each job become known to the online scheduler only after the job is released.

An online scheduler can accomodate dynamic variation on user demand and resource availability; and scheduler make the best use of system resources.

Ex: Priority-driven approach.

———— * ————