Process Scheduling

CPU Scheduling

- ***Basic Concepts***
- ***Scheduling Criteria***
- ***Scheduling Algorithms***

Multilevel Queue Scheduling

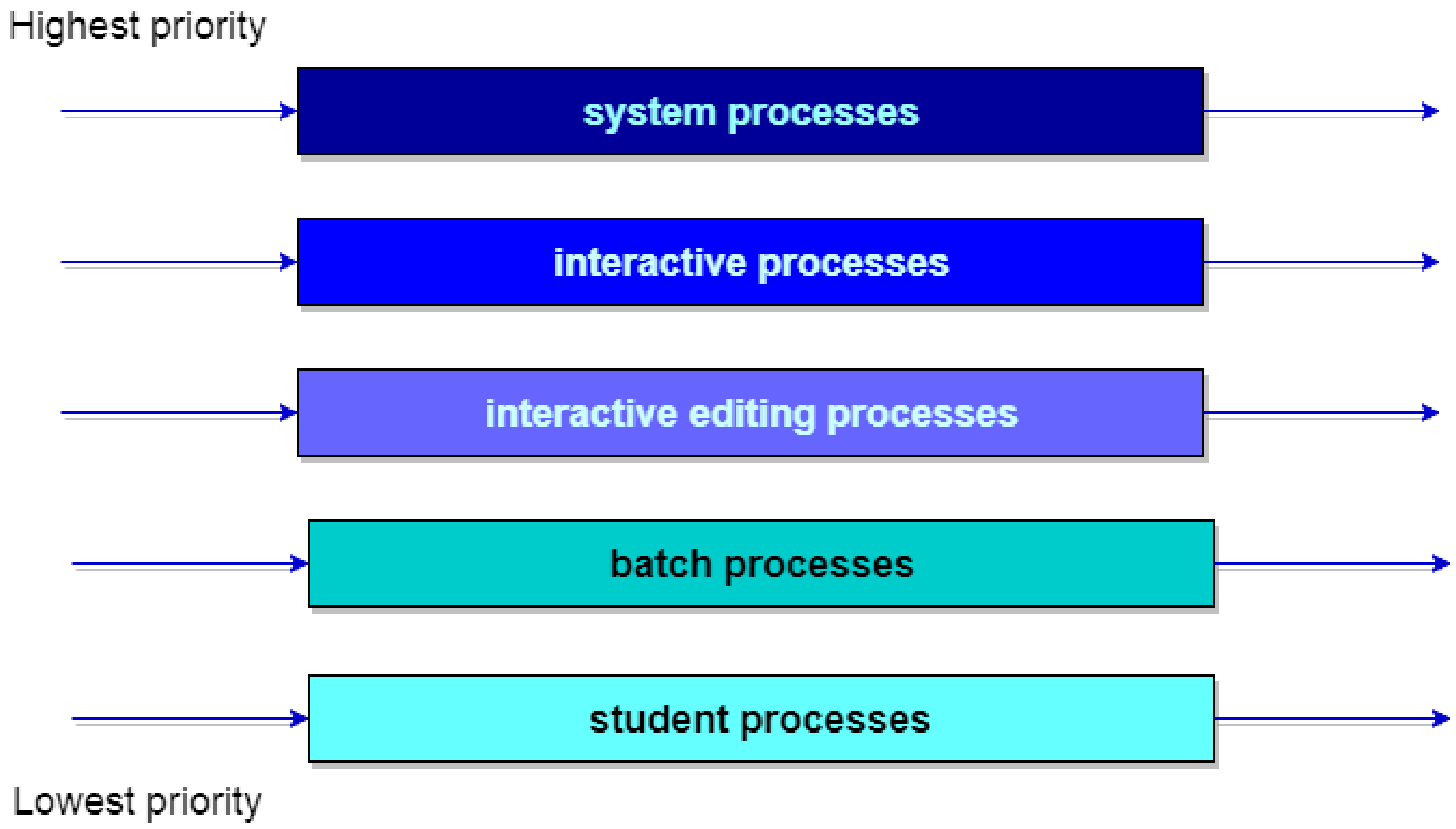
Multilevel Queue Scheduling

- *Multilevel queue scheduling algorithms has been created for situations in which "**processes are easily classified into different groups**". For example a common division is made between **foreground** (interactive) and **background** (batch) processes.*
- *These two process have **different response-time** requirements so may have **different scheduling needs**.*
- *In addition, foreground processes may have priority over background processes.*

Multilevel Queue Scheduling

- *A multilevel queue scheduling algorithm* ***partitions*** *the* ***ready queue*** *into several separate queues.*
- *The processes are* ***permanently*** *assigned to one queue, generally based on some property of the process like memory size, priority or process type. Each queue has its own scheduling algorithm like:* ***foreground – RR, background – FCFS***
- *In addition there must be* ***scheduling among the queue****, which is commonly implemented as* ***fixed priority preemptive scheduling****.* ***Like:*** *For example, the foreground queue may have absolute priority over the background queue.*

Multilevel Queue Scheduling



Multilevel Queue Scheduling

- *Possibility of* ***starvation.***
- ***Another possibility*** *is to time slice among the queues:– Here each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR and 20% to background in FCFS*

Example 1

*Consider below table of **four processes** under **Multilevel queue scheduling**. Queue number denotes the queue of the process. **Priority of queue 1 is greater than queue 2**. **queue 1 uses Round Robin** (Time Quantum = 2) and **queue 2 uses FCFS**. Find out Avg. waiting time and Avg. Turn around time.*

Process	***Arrival Time***	***CPU Burst Time***	***Queue Number***
P1	*0*	*4*	*1*
P2	*0*	*3*	*1*
P3	*0*	*8*	*2*
P4	*10*	*5*	*1*

Queue 1- Ready- Run- TER-

Queue 2-

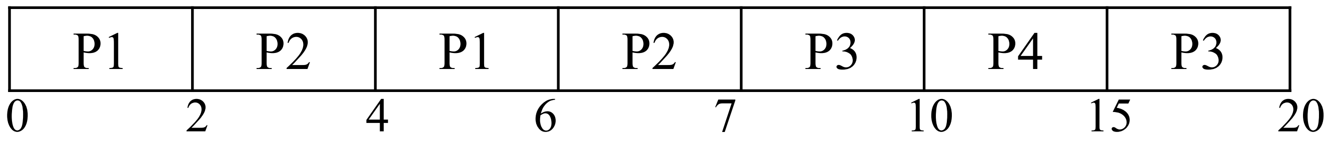
Example 1

*Consider below table of **four processes** under **Multilevel queue scheduling**. Queue number denotes the queue of the process. **Priority of queue 1 is greater than queue 2**. **queue 1 uses Round Robin** (Time Quantum = 2) and **queue 2 uses FCFS**. Find out Avg. waiting time and Avg. Turn around time.*

Process	***Arrival Time***	***CPU Burst Time***	***Queue Number***
P1	*0*	*4*	*1*
P2	*0*	*3*	*1*
P3	*0*	*8*	*2*
P4	*10*	*5*	*1*

Queue 1- EXECUTION- TER-

Queue 2-



Multilevel Feedback Queue Scheduling

Multilevel Feedback Queue Scheduling

In a multilevel feedback queue scheduling algorithm processes are permanently assigned to a queue on entry to the system. Process do not move between the queues. This setup has advantage of low scheduling overhead but disadvantage of being inflexible..

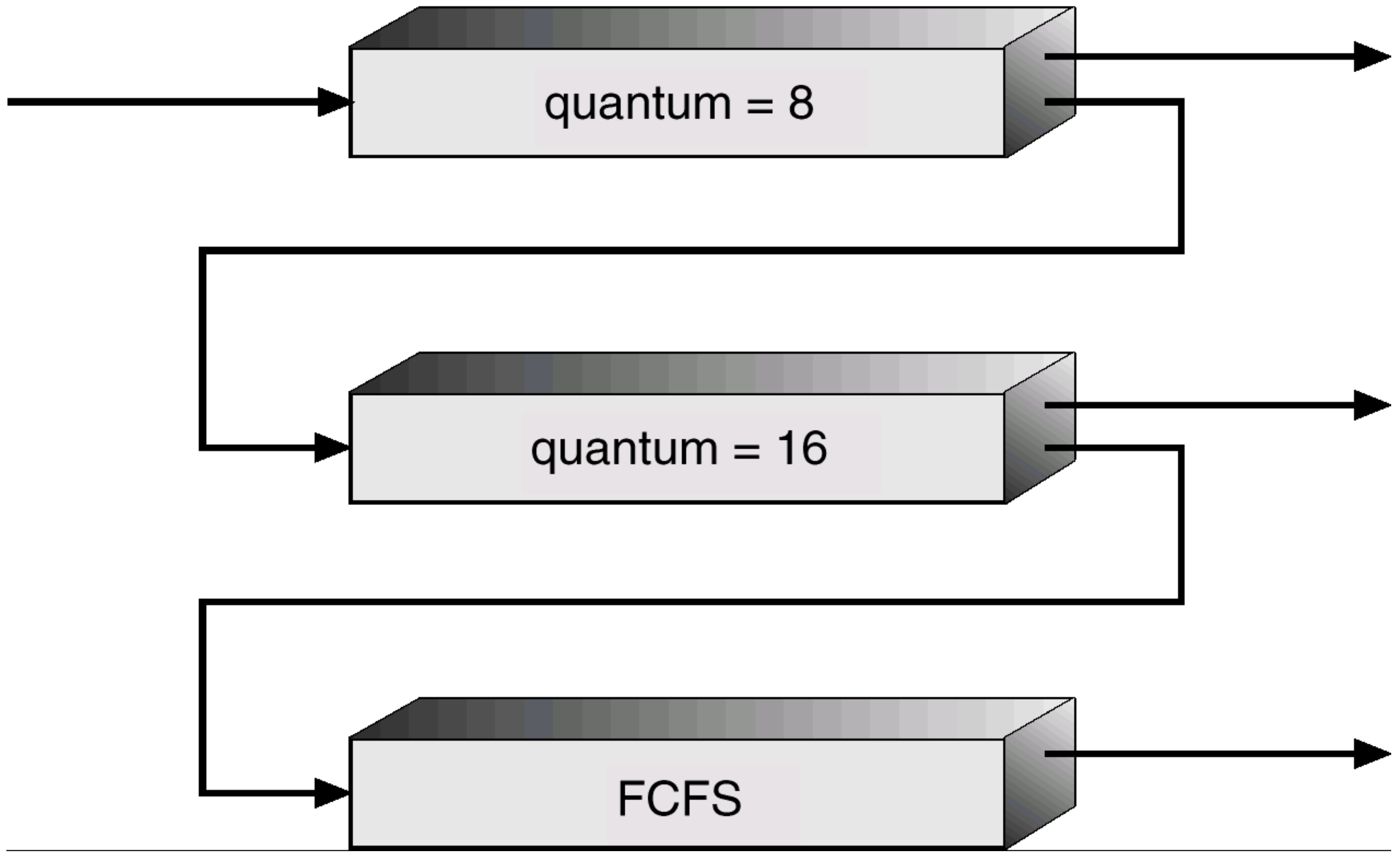
Multilevel Feedback Queue Scheduling

- *Multilevel-feedback-queue scheduling allows a processes to move between the queues.*
- *The idea is to separate processes according to the characteristics of their CPU Burst. If a processes uses too much CPU time it will be moved to the lower priority queue. This leaves I/O bound and Interactive processes in higher priority queue.*
- *In addition, If a process is waiting from too long in lower priority queue may be moved to higher priority queue. This form of aging prevents starvation.*

Multilevel Feedback Queue Scheduling

- *Multilevel feedback queue scheduling keep analyzing the behavior (time of execution) of processes and according to which it changes its priority.*
- *Multilevel-feedback-queue scheduler defined by the following parameters:*
  - *number of queues*
  - *scheduling algorithms for each queue*
  - *method used to determine when to upgrade a process*
  - *method used to determine when to demote a process*
  - *method used to determine which queue a process will enter when that process needs service*

Multilevel Feedback Queue Scheduling



Multilevel Feedback Queue Scheduling

- ***Three queues:***
  - ***Q_0 – time quantum 8 milliseconds***
  - ***Q_1 – time quantum 16 milliseconds***
  - ***Q_2 – FCFS***
- ***Scheduling***
- *A process entering the ready queue is put in queue 0. A process in queue 0 is given a time quantum of **8 milliseconds**.*
- *If it does not finish within this time, it is moved to the tail of queue 1.*
- *If queue 0 is empty, the process at the head of queue 1 is given a quantum of **16 milliseconds.***
- *If it does not complete, it is preempted and is put into queue 2.*
- *Processes in queue 2 are run on an **FCFS basis** but are run only when queues 0 and 1 are empty.*

Example1

Consider a Multilevel Feedback queue Scheduler with three queues numbered from 0 to 2. Show how to schedule the following processes:

Queue 0 = Round Robin (quantum = 17)

Queue 1 = Round Robin (quantum = 25)

Queue 3= FCFS

Process	Burst Time	Arrival Time
P1	53	0
P2	17	0
P3	68	0
P4	24	0

Queue 0- *READY-* *RUN-* *TER-*

Queue 1-

Queue 2-

P1	P2	P3	P4	P1	P3	P4	P1	P3
0	*17*	*34*	*51*	*68*	*93*	*118*	*125*	*136*

Example1

Consider a Multilevel Feedback queue Scheduler with three queues numbered from 0 to 2. Show how to schedule the following processes:

Queue 0 = Round Robin (quantum = 17)

Queue 1 = Round Robin (quantum = 25)

Queue 3= FCFS

Process	Burst Time	Arrival Time
P1	53/ 36/11	0
P2	17/0	0
P3	68/51/26	0
P4	24/7/0	0

Queue 0- *RUN-* *TER-*

Queue 1-

Queue 2-

AWT = (83+17+94+101)/4

AVG TAT = (136+34+162+125)/4

P1	P2	P3	P4	P1	P3	P4	P1	P3

0 17 34 51 68 93 118 125 136

Example 2

Consider below table of four processes under Multilevel Feedback Queue scheduling

.

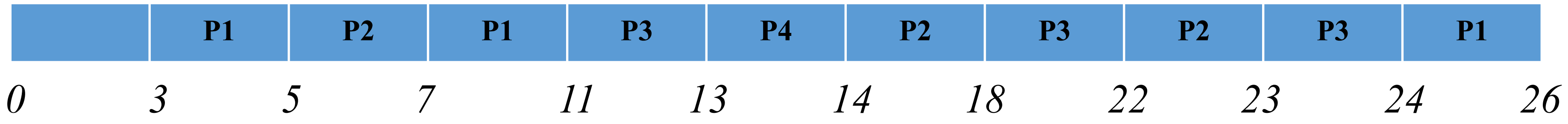
Process	Arrival Time	Burst Time
P1	3	8
P2	4	7
P3	8	7
P4	12	1

queue1 uses round robin scheduling algorithm having time quantum of 2 ms, queue2 also uses round robin scheduling algorithm with time quantum of 4ms and queue3 uses non-preemptive shortest job first algorithm for process scheduling. Priority of queue1 is highest and priority of queue 3 is smallest (queue1>queue2>queue3), and there is non preemptive priority scheduling between the queues. If process waiting time >= 13 ms at any point it must be upgraded to higher priority queue. Find out Avg. waiting time.

Example 2

Process	*Arrival Time*	*Burst Time*
P1	*3*	*8*
P2	*4*	*7*
P3	*8*	*7*
P4	*12*	*1*

Gantt Chart



Example 3

Consider a Multilevel Feedback queue scheduler with three queues, numbered from 0 to 2. Show how to schedule the following processes.

Q_0 - RR Scheduling, Quantum =10
Q_1 - RR Scheduling, Quantum =20
Q_2 - FCFS

Process	Burst Time	Arrival
P_1	12	0
P_2	25	8
P_3	33	21
P_4	2	30

Example 4

Suppose a new process in a system arrives at an average of six processes per minute and each such process requires an average of 8 seconds of service time. Estimate the fraction of time the CPU is busy in a system with a single processor.

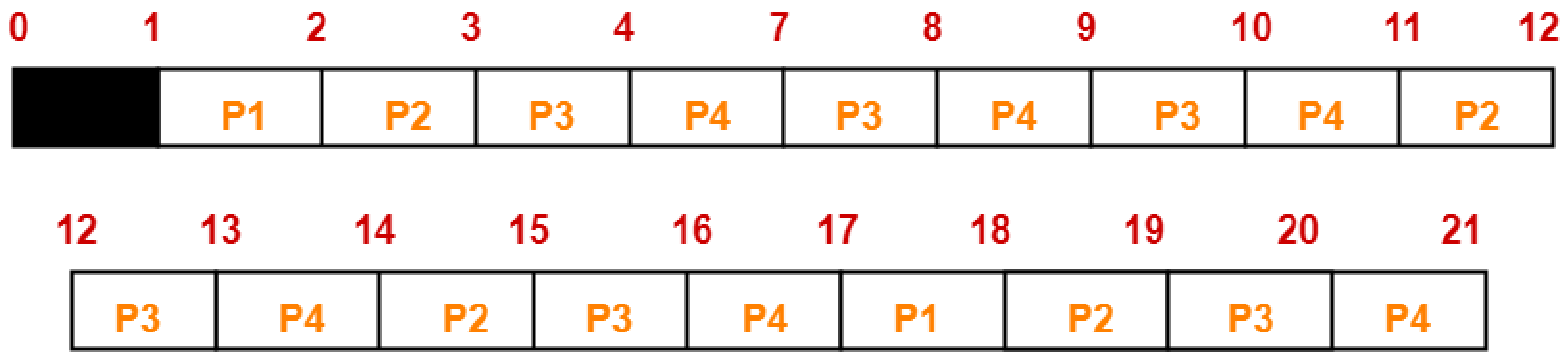
*Ans: 8/10*100*

Example 5

Consider the set of 4 processes whose arrival time and burst time are given below- If the CPU scheduling policy is LJF preemptive, calculate the average waiting time and average turn around time.

Process Id	***Arrival time***	***Burst time***
P1	*1*	*2*
P2	*2*	*4*
P3	*3*	*6*
P4	*4*	*8*

Example- Solution



Gantt Chart

Example 6

Consider three processes (process id 0, 1, 2 respectively) with compute time bursts 2, 4 and 8 time units. All processes arrive at time zero. Consider the longest remaining time first (LRTF) scheduling algorithm. In LRTF ties are broken by giving priority to the process with the lowest process id. The average turn around time is:

13

Example-Solution

```
  p2  p1  p2  p1  p2  p0  p1   p2   p0   p1   p2
0   4   5   6   7   8   9   10    11   12   13   14
```

Turn around time of a process is total time between submission of the process and its completion.

Turn around time of p0 = 12 (12-0)

Turn around time of p1 = 13 (13-0)

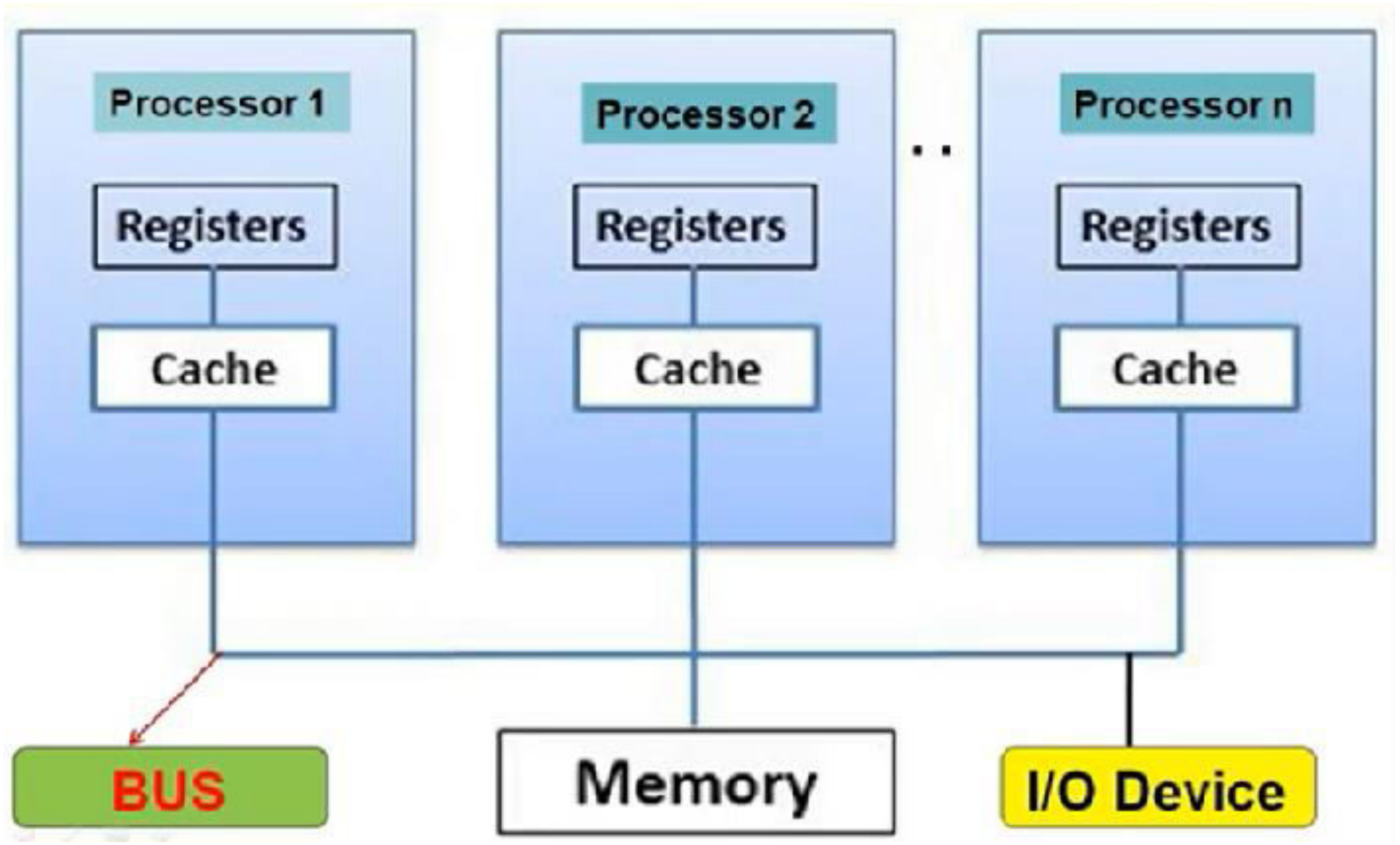
Turn around time of p2 = 14 (14-0)

Average turn around time is (12+13+14)/3 = 13.

Multiple-Processor Scheduling

Multiple-Processor System

A Multi-processor is a system that has more than one processor but shares the ***same memory, bus, and input/output*** *devices.*

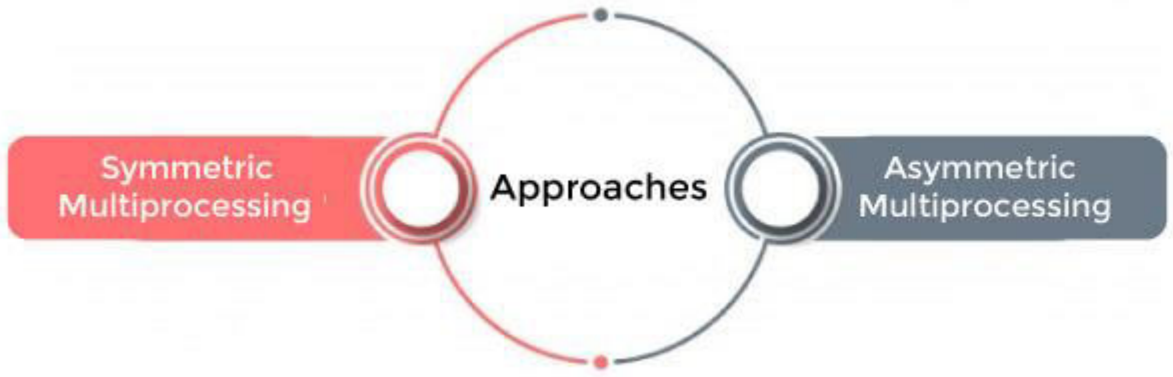


Multiple-Processor System

Multiple-Processor Scheduling

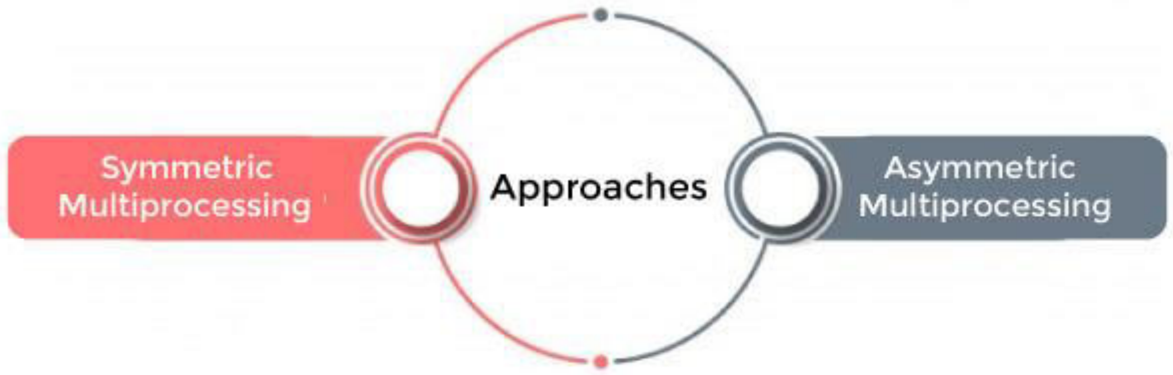
- *Multiple CPUs share the load (load sharing) in multiprocessor scheduling so that various processes run simultaneously.*
- *In general, multiprocessor scheduling is complex as compared to single processor scheduling.*
- *In the multiprocessor scheduling, there are many processors, and they are identical, and we can run any process at any time.*
- *Multiprocessor systems may be heterogeneous (different kinds of CPUs) or homogenous (the same CPU).*

Approaches to Multiple Processor Scheduling



- ***Symmetric Multiprocessing:*** *It is used where each processor is self-scheduling. All processes may be in a common ready queue, or each processor may have its private queue for ready processes. The scheduling proceeds further by having the scheduler for each processor examine the ready queue and select a process to execute.*

Approaches to Multiple Processor Scheduling



- ***Asymmetric Multiprocessing:*** *It is used when all the scheduling decisions and I/O processing are handled by a single processor called the Master Server. The other processors execute only the user code. This is simple and reduces the need for data sharing, and this entire scenario is called Asymmetric Multiprocessing.*

Processor Affinity

Processor affinity *– Most SMP systems try to avoid migration of processes from one processor to another and instead attempt to keep a process running on the same processor. This is known as processor affinity.* ***Process has affinity for the processor on which it is currently running.***

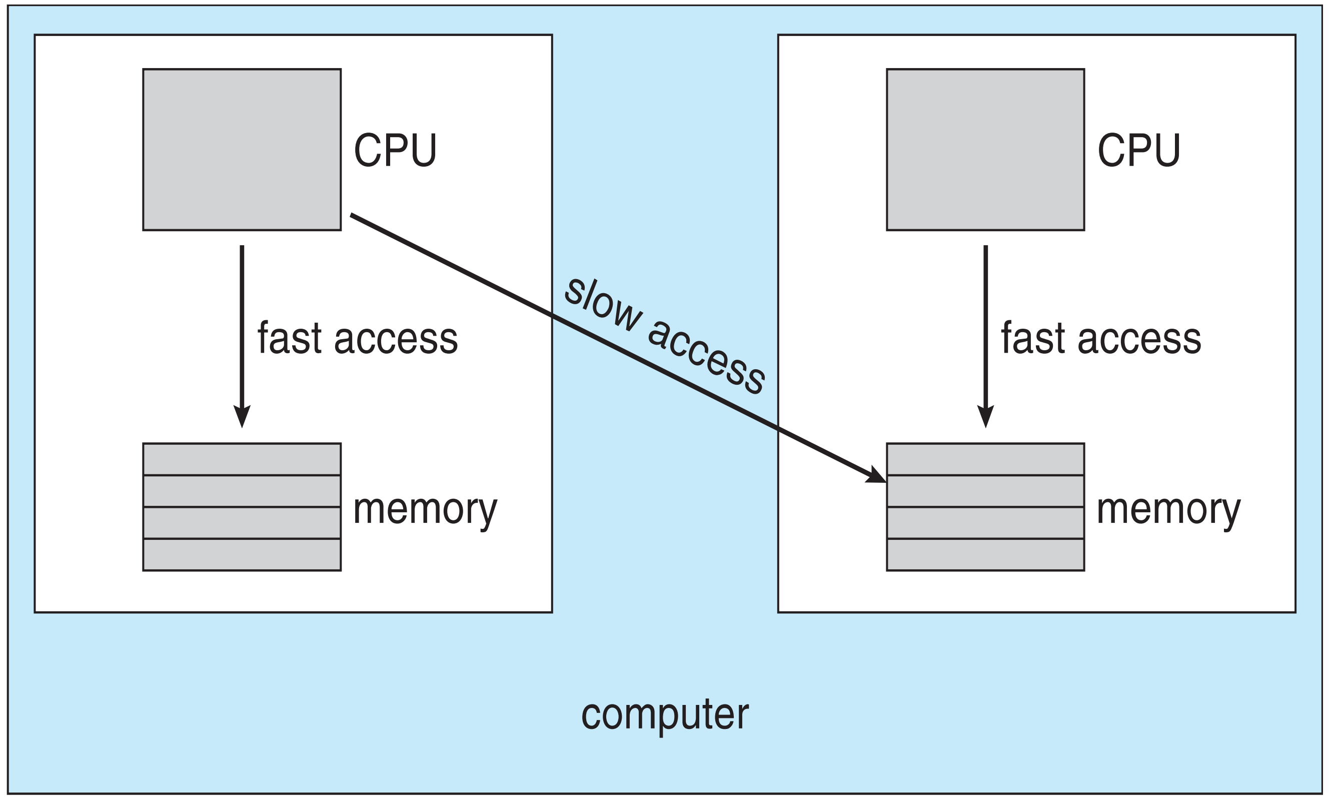
- *The data most recently accessed by the process populate the cache for the processor.*
- *Now, suppose the process migrates to another processor. In that case, the contents of the cache memory must be invalidated for the first processor, and the cache for the second processor must be repopulated.*
- *The high cost involves of invalidating and repopulating caches,*

Types of Processor Affinity

Benefits: The process can take advantage of its data being in that processor's CACHE memory.

- ***Soft Affinity*** *– When an operating system has a policy of attempting to keep a process running on the same processor* ***but not guaranteeing*** *it will do so, this situation is called soft affinity.*
- ***Hard Affinity*** *– Hard Affinity allows a process to specify a* ***subset of processors*** *on which it may run. Some systems such as Linux implements soft affinity but also provide some system calls like sched_setaffinity() that supports hard affinity.*

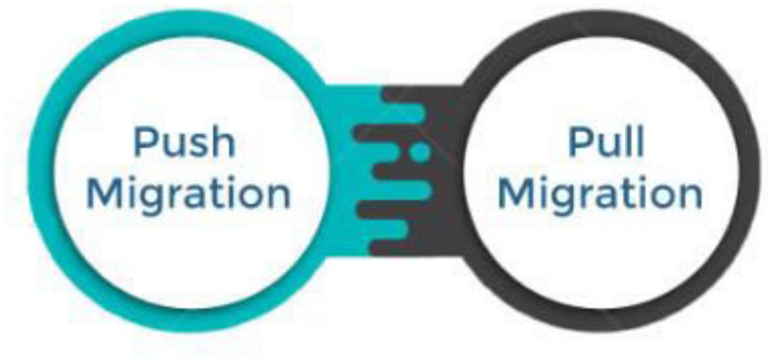
NUMA and CPU Scheduling



Note that memory-placement algorithms can also consider affinity

Load Balancing

- *Load Balancing is the phenomenon that keeps the workload evenly distributed across all processors in an SMP system.*



- ***Push Migration** – In push migration a task routinely checks the load on each processor and if it finds an imbalance then it evenly distributes load on each processors by moving the processes from overloaded to idle or less busy processors.*
- ***Pull Migration** – Pull Migration occurs when an idle processor pulls a waiting task from a busy processor for its execution.*