Memory Management cont…

The memory management subsystem is one of the most important parts of the operating system. Since the early days of computing, there has been a need for more memory than exists physically in a system. Strategies have been developed to overcome this limitation and the most successful of these is virtual memory.

Virtual memory

makes the system appear to have more memory than is physically present by sharing it among competing processes as they need it. Virtual memory does more than just make your computer's memory go farther. The memory management subsystem provides:

Virtual memory

makes the system appear to have more memory than is physically present by sharing it among competing processes as they need it. Virtual memory does more than just make your computer's memory go farther. The memory management subsystem provides:

Large Address Spaces

The operating system makes the system appear as if it has a larger amount of memory than it actually has. The virtual memory can be many times larger than the physical memory in the system.

Protection

Each process in the system has its own virtual address space. These virtual address spaces are completely separate from each other and so a process running one application cannot affect another. Also, the hardware virtual memory mechanisms allow areas of memory to be protected against writing. This protects code and data from being overwritten by rogue applications

Virtual memory …

Memory Mapping

Memory mapping is used to map image and data files into a process' address space. In memory mapping, the contents of a file are linked directly into the virtual address space of a process.

Fair Physical Memory Allocation

The memory management subsystem allows each running process in the system a fair share of the physical memory of the system.

Shared Virtual Memory

Although virtual memory allows processes to have separate (virtual) address spaces, there are times when you need processes to share memory.

For example

There could be several processes in the system running the bash command shell.
Rather than have several copies of bash, one in each process's virtual address space, it is better to have only one copy in physical memory and all of the processes running bash share it.

Virtual memory …

Dynamic libraries are another common example of executing code shared between several processes.

Shared memory can also be used as an Inter Process Communication (IPC) mechanism, with two or more processes exchanging information via memory common to all of them.

Linux supports the Unix System V shared memory IPC

PROCESS SCHEDULING

CPU is always busy in **Multiprogramming**. Because CPU switches from one job to another job. But in **simple computers** CPU sit idle until the I/O request granted.

scheduling is a important OS function. All resources are scheduled before use.(cpu, memory, devices…..)

Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing

PROCESS SCHEDULING:

CPU is always busy in **Multiprogramming**. Because CPU switches from one job to another job. But in **simple computers** CPU sit idle until the I/O request granted.

scheduling is a important OS function. All resources are scheduled before use.(cpu, memory, devices…..)

Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing

Context Switch: *Assume, main memory contains more than one process. If cpu is executing a process, if time expires or if a high priority process enters into main memory, then the scheduler saves information about current process in the PCB and switches to execute the another process. The concept of moving CPU by scheduler from one process to other process is known as context switch.*

Non-Preemptive Scheduling: *CPU is assigned to one process, CPU do not release until the competition of that process. The CPU will assigned to some other process only after the previous process has finished.*

Preemptive scheduling: here CPU can release the processes even in the middle of the execution. CPU received a signal from process p2. OS compares the priorities of p1 ,p2. If p1>p2, CPU continues the execution of p1. If p1<p2 CPU preempt p1 and assigned to p2.

Dispatcher: The main job of dispatcher is switching the cpu from one process to another process. Dispatcher connects the cpu to the process selected by the short term scheduler.

Dispatcher Latency: The time it takes by the dispatcher to stop one process and start another process is known as dispatcher latency. If the dispatcher latency is increasing, then the degree of multiprogramming decreases.

SCHEDULING CRITERIA

1. Throughput: how many jobs are completed by the CPU with in a time period.

2. Turn around time : The time interval between the submission of the process and time of the completion is turn around time.

*TAT = **Waiting time in ready queue + executing time + waiting time in waiting queue for I/O.***

3. Waiting Time: The time spent by the process to wait for cpu to be allocated.

4. Response Time: Time duration between the submission and first response.

5. CPU Utilization: CPU is costly device, it must be kept as busy as possible.

Eg: CPU efficiency is 90% means it is busy for 90 units, 10 units idle.

CPU SCHEDULINGALGORITHMS

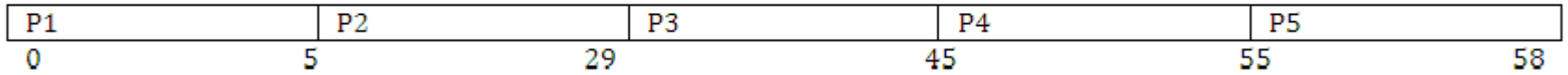
1. First come First served scheduling: (FCFS): The process that request the CPU first is holds the cpu first. If a process request the cpu then it is loaded into the ready queue, connect CPU to that process.

Consider the following set of processes that arrive at time 0, the length of the CPU burst time given in milli seconds

Process	Burst time(milliseconds)
P1	5
P2	24
P3	16
P4	10
P5	3

burst time is the time, required the CPU to execute that job, it is in milli seconds.

Chart:



Average turn around time

Turn around time= waiting time + burst time

- ✓ Turn around time for p1= 0+5=5.
- ✓ Turn around time for p2=5+24=29
- ✓ Turn around time for p3=29+16=45
- ✓ Turn around time for p4=45+10=55
- ✓ Turn around time for p5= 55+3=58

Average turn around time= (5+29++45+55+58/5) = 187/5 =37.5 millisecounds

Average waiting time

waiting time= starting time- arrival time

Waiting time for p1=0
Waiting time for p2=5-0=5
Waiting time for p3=29-0=29
Waiting time for p4=45-0=45
Waiting time for p5=55-0=55

Average waiting time= 0+5+29+45+55/5 = 125/5 = 25 ms.

Average Response Time

Formula : First Response - Arrival Time

- ✓ Response Time for P1 =0
- ✓ Response Time for P2 => 5-0 = 5
- ✓ Response Time for P3 => 29-0 = 29
- ✓ Response Time for P4 => 45-0 = 45
- ✓ Response Time for P5 => 55-0 = 55

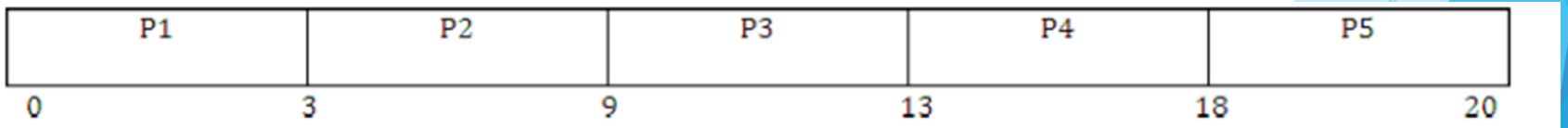
Average Response Time => (0+5+29+45+55)/5 =>25ms

First Come First Serve

It is Non Primitive Scheduling Algorithm

PROCESS	BURST TIME	ARRIVAL TIME
P1	3	0
P2	6	2
P3	4	4
P4	5	6
P5	2	8

P1 arrived at 0 ms.
P2 arrived at 2 ms.
P3 arrived at 4 ms.
P4 arrived at 6 ms.
P5 arrived at 8 ms.



Average Turn Around Time

Formula : ***Turn around Time =: waiting time + burst time***

- ✓ Turn Around Time for P1 => 0+3= 3
- ✓ Turn Around Time for P2 => 1+6 = 7
- ✓ Turn Around Time for P3 => 5+4 = 9
- ✓ Turn Around Time for P4 => 7+ 5= 12
- ✓ Turn Around Time for P5 => 2+ 10=12

Average Turn Around Time => (3+7+9+12+12)/5 =>43/5 = 8.50 ms.

Average Response Time

Formula : ***Response Time = First Response - Arrival Time***

- ✓ Response Time of P1 = 0
- ✓ Response Time of P2 => 3-2 = 1
- ✓ Response Time of P3 => 9-4 = 5
- ✓ Response Time of P4 => 13-6 = 7
- ✓ Response Time of P5 => 18-8 =10

Average Response Time => (0+1+5+7+10)/5 => 23/5 = 4.6 ms

Advantages

- ✓ Easy to Implement,
- ✓ Simple

Disadvantage

Average waiting time is very high.

Other Algorithms include:-

Shortest Job First Scheduling (SJF)

Shortest Remaining Time First (SRTF)

ROUND ROBIN SCHEDULING ALGORITHM

RATE MONOTONIC (RM) SCHEDULING ALGORITHM

Starvation means only high priority process are executing, but low priority process are waiting for the CPU for the longest period of the time.

Multiple – processor scheduling

When multiple processes are available, then the scheduling gets more complicated, because there is more than one CPU which must be kept busy and in effective use at all times.

Load sharing resolves around balancing the load between multiple processors. Multi processor systems may be heterogeneous (It contains different kinds of CPU's) (or) Homogeneous(all the same kind of CPU).

1) Approaches to multiple-processor scheduling

a)Asymmetric multiprocessing

One processor is the master, controlling all activities and running all kernel code, while the other runs only user code.

b)Symmetric multiprocessing:

Each processor schedules its own job. Each processor may have its own private queue of ready

2. Processor Affinity

Successive memory accesses by the process are often satisfied in cache memory. what happens if the process migrates to another processor. the contents of cache memory must be invalidated for the first processor, cache for the second processor must be repopulated. Most Symmetric multi processor systems try to avoid migration of processes from one processor to another processor, keep a process running on the same processor. This is called processor affinity.

a) Soft affinity:

Soft affinity occurs when the system attempts to keep processes on the same processor but makes no guarantees.

b) Hard affinity:

Process specifies that it is not to be moved between processors

3) Load balancing:

One processor wont be sitting idle while another is overloaded. Balancing can be achieved through push migration or pull migration.

Push migration:

Push migration involves a separate process that runs periodically(e.g every 200 ms) and moves processes from heavily loaded processors onto less loaded processors.

Pull migration:

Pull migration involves idle processors taking processes from the ready queues of the other processors.

Real time scheduling

Real time scheduling is generally used in the case of multimedia operating systems. Here multiple processes compete for the CPU. How to schedule processes A,B,C so that each one meets its deadlines. The general tendency is to make them pre-emptable, so that a process in danger of missing its deadline can preempt another process. When this process sends its frame, the preempted process can continue from where it had left off. Here throughput is not so significant. Important is that tasks start and end as per their deadlines.

RATE MONOTONIC (RM) SCHEDULING ALGORITHM

Rate monotonic scheduling Algorithm works on the principle of preemption. Preemption occurs on a given processor when higher priority task blocked lower priority task from execution. This blocking occurs due to priority level of different tasks in a given task set. rate monotonic is a preemptive algorithm which means if a task with shorter period comes during execution it will gain a higher priority and can block or preemptive currently running tasks. In RM priorities are assigned according to time period. Priority of a task is inversely proportional to its timer period. Task with lowest time period has highest priority and the task with highest period will have lowest priority.

Feedback/Complain

Your feedback on all aspects of this course is welcome. Feedback allows any problems arising to be fixed immediately rather than compounding it and making things difficult

Sir, Abubakar UMAR
Department of Mathematical Sciences,
Computer Science Option.
Office Location: NEEDS ASSESSMENT BUILDING OPP. CHURCH (2ND TO THE LAST OFFICE),
Yelwa Campus, BAUCHI. BAUCHI STATE
abkamir@gmail.com
+2348034456198 (Whatapp only)