COMPUTER NETWORK (CS-303)

TUTORIAL 4

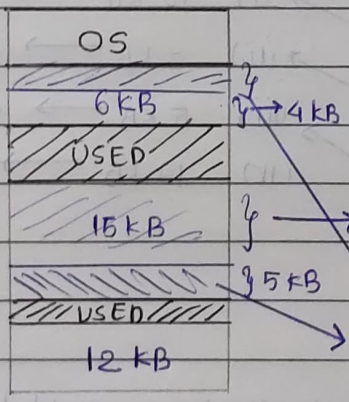
[BHAGYA VINOD RANA]
U19CS012

1) Select the correct alternative in each of following questions:

a.) A worst-fit allocator always split the largest free memory area while making an allocation. A free list contains three memory areas of sizes 6KB, 15KB, and 12KB. The next four memory requests are for 10KB, 2KB, 5KB and 14 KB of memory. The only placement strategy that would be able to accomodate all four processes is:

(i) First-fit (X)
(ii) ✓ Best-fit
(iii) Worst-fit (X)
(iv) Next-fit (X)

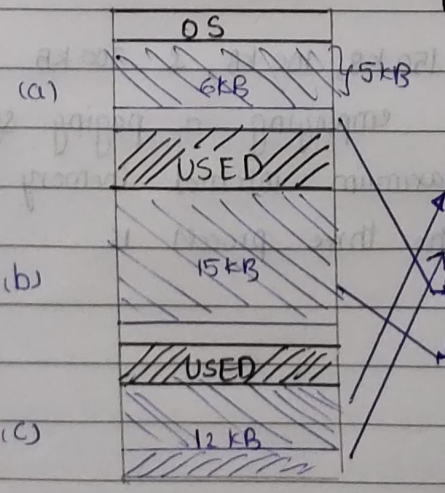
(a)



First Fit - first sufficient block from top of main memory (first hole that is large enough)

① 10KB → (b)
② 2KB → (a)
③ 5KB → (b)
④ 14 KB → (X) can't be accomodated

(b) (ii)



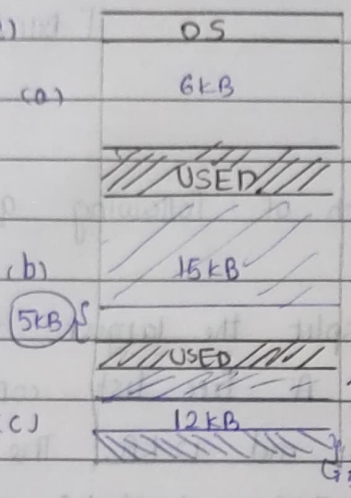
Best Fit - first smallest sufficient partition among free available partition.
(smallest hole (>= size of process))

① 10 KB → (c)
② 2KB → (c)
③ 5KB → (a)
④ 14 KB → (b)

∵ Best Fit is able to accomodate all processes
∴ Option (ii) Best Fit is our Answer

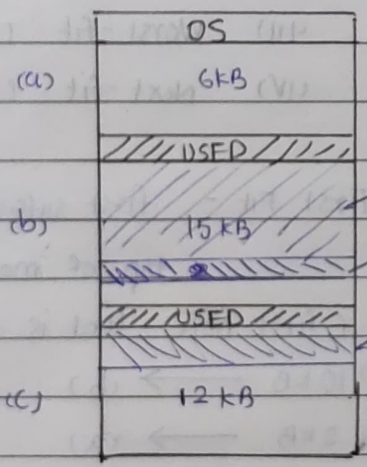
U19CS012

(c) Worst Fit - Allocate the process → sufficiently large (opposite to best fit) (tag Largest hole → process)



(i) 10 kB → (b)
(ii) 2 kB → (c)
(iii) 5 kB → (c)
(iv) 14 kB → ⊗ can't be accomodated.

(d) Next Fit - similar to first fit but it will search for first sufficient partition from last allocation file.



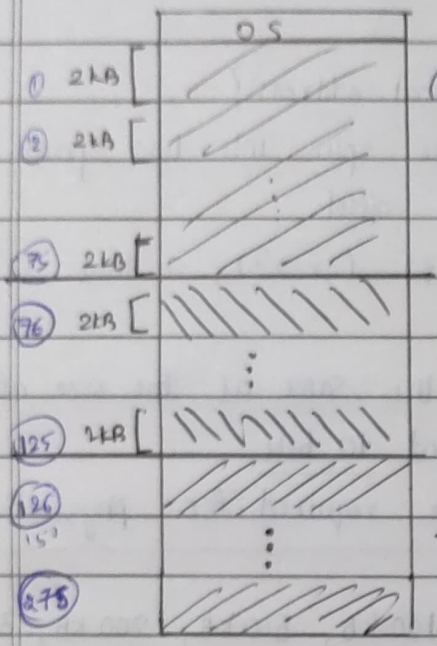
(i) 10 kB → (b)
(ii) 2 kB → (b)
(iii) 5 kB → (c)
(iv) 14 kB → ⊗ can't be accomodated.

Ans: (ii) Best Fit placement strategy would be able to accomodate all 4 places.

(b) Three process requiring 150 kB, 100 kB, & 300 kB of memory are in operation in an OS employing a paging system with a page size of 2 kB. The maximum internal memory fragmentation due to memory allocation to the three process is

(i) Approximately 2 kB
(ii) Approximately 6 kB
(iii) 275 kB
(iv) None of (i) - (iii)

UI9CS012



(1-75 Blocks)

(76-125 Block No)

(126-275 Block No)

Blocks

① 150 KB → (75) × 2 KB = 150 KB

Blocks

② 100 KB → (50) × 2 KB = 100 KB

Blocks

③ 300 KB → (150) × 2 KB = 300 KB

Internal Fragmentation = 0 KB + 0 KB + 0 KB
= 0

Ans: (iv) None of (i) - (iii) { 0 KB = No internal Fragmentation }

(c) A reentrant program is one that
(i) Calls itself recursively
(ii) Can have several copies in memory than can be used by different users
(iii) ✓ Can have single copy in memory that is executed by many users concurrently.

Ans: A ~~re-entrant~~ computer program or subroutine is called reentrant if multiple invocations can safely run concurrently on single processor system.
∴ re-entrant program is one that (iii) can have single copy in memory that is executed by many users concurrently.

2.) Consider a logical address space of 64 pages of 1024 words, each mapped onto a physical memory of 32 frames.

a.) How many bits are there in the logical address?

The Logical address is split into two parts = The page address & offset

① The page address must be able to reference 64 (or 2^6) distinct pages. ↳ requires 6 address bit.

② The offset must be able to reference 1024 (or 2^{10}) distinct words on each page. ↳ requires 10 address bit.

Therefore, 16 (6+10) Bits are required for Logical address

(b) How many bits are in physical address?

① The physical address is also split into two parts - Frame address and the offset

② The number of frames is 32 (or 2^5), so that will require 5 bits.

③ The size of the frame is the same as the size of the page, therefore this will required 10 bits.

Therefore, 15 bits {5+10} are required for physical address.

3) Given five memory partitions of 100 kB, 500 kB, 200 kB, 300 kB, and 600 kB. (in order) How would the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 kB and 426 KB Which algorithm makes the most efficient use of memory. (in order)?

3)

a) First-Fit

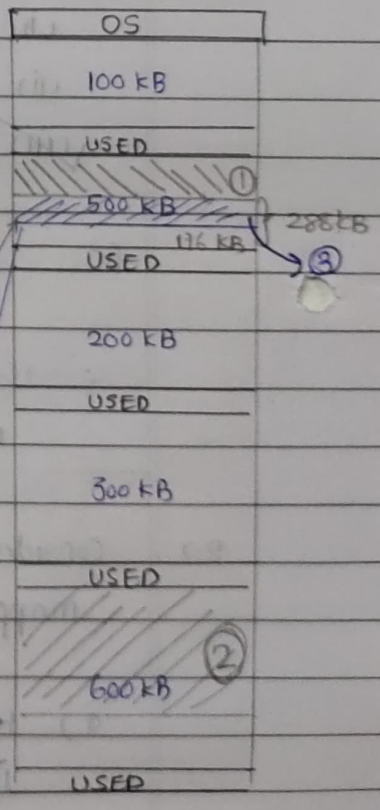
The First Fit algorithm selects the first free partition that is large enough to accomodate the request.

① 212 kB ⇒ 500 kB partition, leaves 288 kB

② 417 kB ⇒ 600 kB partition, leaves 183 kB

③ 112 kB ⇒ 288 kB partition, leaves 176 kB

④ 426 kB would not be able to allocate, no partition large enough.

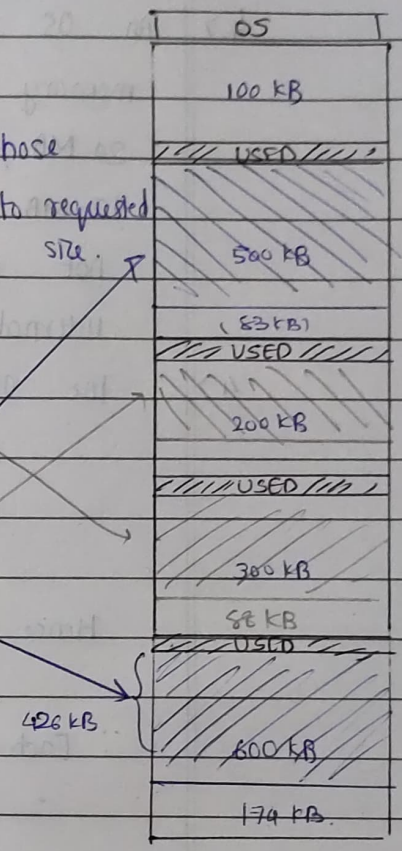


(b) Best Fit

The best fit algorithm selects the partition whose size is closest in size (and large enough) to requested size.

Best Fit would allocated in following manner:

① 212 KB ⇒ 300 KB, leaving 88 KB
② 417 KB ⇒ 500 KB, leaving 83 KB
③ 112 KB ⇒ 200 KB, leaving 88 KB
④ 426 KB ⇒ 600 KB, leaving 174 KB

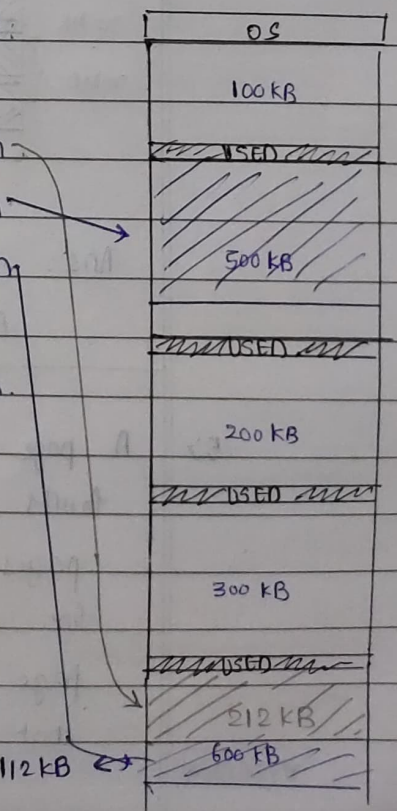


(c) Worst Fit

The worst fit algorithm effectively selects the largest partition for each request.

Worst-fit would allocate in the following manner:

① 212 KB ⇒ 600 KB, leaving 388 KB partition
② 417 KB ⇒ 500 KB, leaving 83 KB partition
③ 112 KB ⇒ 388 KB, leaving 276 KB partition
④ 426 KB ⇒ would not be allowed to allocate as no partition is large enough.



(d) Which algorithm makes the most effecient use of memory?

The Best-fit algorithm performed the best of the three algorithm, as it was the only algorithm to meet all memory requests.

U19CS012

4.) An OS has 110 MB available for user processes. The maximum memory requirement of a process for its own code and data is 20 MB, while the average memory requirement of a process is 10 MB. If the OS uses contigous memory allocation and does not know size of individual processes, what is the average internal and external fragmentation?

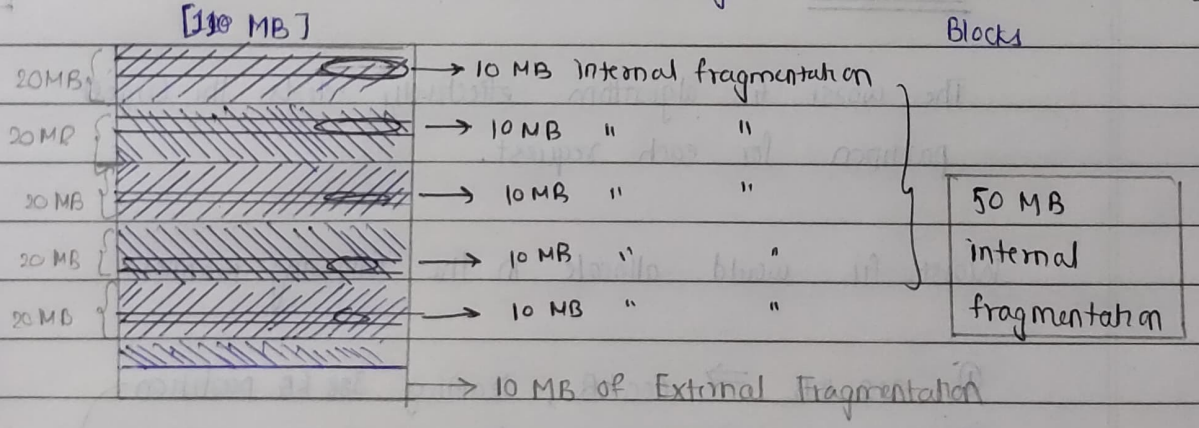
4.) The OS creates 5 memory partitions of 20 MB each.

$$110\ MB = [5 \times 20\ MB + 10\ MB \text{ in next Block}]$$

Hence, External Fragmentation is **10 MB**.

∵ Each partition has average internal fragmentation of 10 MB.

∴ Hence the total internal fragmentation = $5 \times 10\ MB = 50\ MB$.



Ans: Average internal Fragmentation = 50 MB

Average external Fragmentation = 10 MB.

5.) A page replacement algorithm should minimize the number of page faults. We can acheive this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. We can associate with each page frame a counter of the number of pages associated with that frame.

Then, to replace a page, we can search for page frame with the smallest counter.

STEP 1 : The Algorithm [Define a page replacement algorithm using basic idea]

The algorithm minimizes number of page faults by distributing heavily used pages evenly over all of memory, rather than having them compete for small number of page frames.

(a) Address below problems

(i) What is the initial value of counters?

The initial values of the counter is 0.

(ii) When are counters increased?

When a new page is associated with the frame, its counter is increased by one.

(iii) When are counters decreased?

When a page is associated with the frame, ~~its counter is~~ is no longer required in its counter is decreased by one.

(iv) How is the page to be replaced selected?

The page frame with the smallest counter is replaced. However, if when tie occurs, the FIFO algorithm will be used.

(b) How many page faults occur for your algorithm for the following reference string with four page frames?

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2

① Page 1, 2, 3 and 4 are filled to all 4 frames counters are 1.

② Page 5 moved to Frame-1, based on FIFO algorithm, its counter is 2 now.

U19CS012

③ Page 3 and 4 are available, so no fault occurs and counter remains same.

④ Page 1 moved to Frame 2 based on FIFO algorithm and its counter is changed to 2.

⑤ Page 6 is moved to Frame-3 based on FIFO algorithm. and its counter is changed to 2. But since 3 is not required later on, its counter is again decreased by 1. So, its counter is now 1 again.

⑥ Page 7 is moved to Frame 4 based on FIFO algorithm and its counter is changed to 2.

ELEMENT	1	2	3	4	5	3	4	1	6	7	8	7	8	9	7	8	9	5	4	5	4	2
FRAME 1	1(1)				5(2)											8(3)						
FRAME 2		2(1)						1(2)										3-1=2 5(2)		NF		
FRAME 3			3(1)			NF		1(2)	2-1=1 6(1)		2-1=1 8(1)		NF	9(2)			NF					
FRAME 4				4(1)			NF			7(2)		NF			NF				2-1=1 4(1)		NF	2

FAULTS = 22-8 = 14 Page Faults

⑦ Page 8 is moved to Frame 3 on new algorithm (its counter is lowest) and its counter is changed to 2. But since 6 is not required later on, its counter is again decreased by 1. So, its counter is now 1 again.

⑧ Page 7 and Page 8 are available in frames 3 and 4, so not fault occurs. and counter remains same.

⑨ Page 9 is moved to frame 3 based on new algorithm (its counter is lowest) and its counter is changed to 2.

⑩ All the counters are 2 now, so Page 8 is moved to frame 1 with FIFO algorithm and its counter is 3 now.

⑪ Page 9 is available in frame 3, so no fault occurs and

Counter remains the same.

(12) Page 5 is moved to frame-2 based on FIFO algorithm and counter is changed to 2. However, 1 will not be required again, so it is changed back to 1.

(13) Page 4 is moved to frame 4 based on FIFO Algorithm and counter is changed to 2. However, 1 will not be required again, so it is changed to 1 back.

(14) Pages 5 and 4 are available in frames 2 and 3 so no fault occurs and counter remains same.

(15) Page 2 is moved to frame-4 based on new algorithm.

(c) What is the minimum number of page faults for an optimal page replacement strategy for the reference string in part b with four page frames.

The optimal algorithm is shown below:-

ELEMENT	1	2	3	4	5	3	4	1	6	7	8	7	8	9	7	8	9	5	4	5	4	2
FRAME 1	1							NF	6		8		NF			NF						
FRAME 2		2			5													NF		NF		2
FRAME 3			3			NF				7		NF			NF				4		NF	
FRAME 4				4			NF							9			NF					

Faults = 22 - 11 = 11 ← Ans

Ans: Result (a) Works based on counter

(b) 14 page Faults

(c) 11 page Faults.

U19CS012

6.) Consider the following page reference string:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six and seven frames? Remember, all frames are initially empty, so your first unique pages will cast one fault each.

(A) LRU replacement (B) FIFO replacement (C) Optimal replacement

(I) LRU: Frames = 1

Consider the least recently used method with one frame:

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME-1	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6

FAULTS = 20

(II) LRU: Frames = 2

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1		3		2		5		2		NF		7		3		1		3	
FRAME 2		2		4		1		6		1		3		6		2		NF		6

FAULTS = 20 - 2 = 18

(III) LRU: Frames = 3

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1			4			5			1			7			2		NF		
FRAME 2		2			NF			6				3			NF				NF	
FRAME 3			3			1			2		NF			6			1			6

FAULTS = 20 - 5 = 15

U19CS012

(IV) LRU : Frames = 4

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1					NF				NF				6						NF
FRAME 2		2			NF				NF		NF					NF		NF		
FRAME 3			3				5					3			NF				NF	
FRAME 4				4				6					7				1			

FAULTS = 20 - 10 = 10

(V) LRU : Frames = 5

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1					NF				NF							NF			
FRAME 2		2			NF				NF		NF					NF		NF		
FRAME 3			3					6						NF						NF
FRAME 4				4								3			NF				NF	
FRAME 5							5						7							

FAULTS = 20 - 12 = 8

(VI) LRU : Frames = 6

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1					NF				NF							NF			
FRAME 2		2			NF				NF		NF					NF		NF		
FRAME 3			3									NF			NF				NF	
FRAME 4				4																
FRAME 5							5						7							
FRAME 6								6						NF						NF

FAULTS = 20 - 13 = 7

VI9CS012

(VII) LRU : Frames = 7

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame 1	1					NF				NF							NF			
Frame 2		2			NF				NF		NF					NF		NF		
Frame 3			3									NF			NF				NF	
Frame 4				4																
Frame 5							5													
Frame 6								6						NF						NF
Frame 7													7							

FAULTS = 20 - 13 = 7

(I) FIFO : Frames = 1

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame 1	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6

FAULTS = 20

(II) FIFO : Frames = 2

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame 1	1		3		2		5		2		NF		7		3		1		3	
Frame 2		2		4		1		6		1		3		6		2		NF		6

FAULTS = 20 - 2 = 18

(III) FIFO : Frames = 3

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame 1	1			4				6				3			NF	2		NF		6
Frame 2		2			NF	1			2		NF		7				1			
Frame 3			3				5			1				6					3	

FAULTS = 20 - 4 = 16

vision

U19CS012

(IV) FIFO = Frames = 4

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame 1	1					NF	5					3			NF		1			
Frame 2		2			NF			6					7						3	
Frame 3			3						2		NF			6						NF
Frame 4				4						1						2		NF		

FAULTS = 20 - 6 = 14

(V) FIFO = Frames = 5

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame 1	1					NF		6						NF						NF
Frame 2		2			NF				NF	1							NF			
Frame 3			3								2				NF			NF		
Frame 4				4								3			NF				NF	
Frame 5							5						7							

FAULTS = 20 - 10 = 10

(VI) FIFO = Frames = 6

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame 1	1					NF				NF			7							
Frame 2		2			NF				NF		NF					NF	1			
Frame 3			3									NF			NF			2		
Frame 4				4															3	
Frame 5							5													
Frame 6								6						NF						NF

FAULTS = 20 - 10 = 10

Next Page →

U19C3012

(VII) FIFO : Frames = 7

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1					NF				NF							NF			
FRAME 2		2			NF				NF		NF					NF		NF		
FRAME 3			3									NF			NF				NF	
FRAME 4				4																
FRAME 5							5													
FRAME 6								6						NF						NF
FRAME 7													7							

FAULTS = 20 - 13 = 7

(I) Optimal : Frames = 1

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6

FAULTS = 20

(II) Optimal : Frames = 2

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1		3	4		1	5	6		1		3	7	6	3		1		3	6
FRAME 2		2			NF				NF		NF					NF		NF		

FAULTS = 20 - 5 = 15

(III) Optimal : Frames = 3

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1					NF				NF		3			NF				NF	
FRAME 2		2			NF				NF		NF		7			2		NF		
FRAME 3			3	4			5	6						NF			1			6

FAULTS = 20 - 9 = 11

vision

U19CS012

(IV) Optimal: Frames = 4

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1					NF				NF			7				1			
FRAME 2		2			NF				NF		NF					NF		NF		
FRAME 3			3									NF			NF				NF	
FRAME 4				4			5	6						NF						NF

FAULTS = 20 - 12 = 8

(V) Optimal: Frames = 5

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1					NF				NF							NF			
FRAME 2		2			NF				NF		NF					NF		NF		
FRAME 3			3									NF			NF				NF	
FRAME 4				4									7							
FRAME 5							5	6						NF						NF

FAULTS = 20 - 13 = 7

(VI) Optimal: Frames = 6

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME 1	1					NF				NF							NF			
FRAME 2		2			NF				NF		NF					NF		NF		
FRAME 3			3									NF			NF				NF	
FRAME 4				4																
FRAME 5							5						7							
FRAME 6								6						NF						NF

FAULTS = 20 - 13 = 7

U19CS012

(VII) Optimal : Frames = 7

ELEMENT	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
FRAME-1	1					NF				NF							NF			
FRAME-2		2			NF				NF		NF					NF		NF		
FRAME-3			3									NF		NF					NF	
FRAME-4				4																
FRAME-5							5													
FRAME-6								6						NF						NF
FRAME-7													7							

FAULTS = 20-13 = 7

Result

	No. of Faults		
NO OF FAULTS	LRU	FIFO	Optimal
FRAMES-1	20	20	20
FRAME-2	18	18	15
FRAME-3	15	16	11
FRAME-4	10	14	8
FRAME-5	8	10	7
FRAME-6	7	10	7
FRAME-7	7	7	7

7.) Consider a demand paged computer system where the degree of multiprogramming is currently fixed at four. The system was recently measured to determine utilization of the CPU and the paging disk. The results are one of the following alternatives.
For each case, what is happening? CPU utilization?
Can degree of multiprogramming be increase to increase
Is the paging helping?

U19CS012

* The degree of multiprogramming - It is the maximum number of processes that a single processor system can accomodate effeciently.

* Thrashing - Thrashing is a condition or a situation when the system is spending a major portion of its time in servicing the page faults, but the actual processing done is very negligible.

(a) CPU utilization = 13% & Disk utilization = 97%

Disk utilization at 97% is too high while the CPU at 13% is too low, this causes thrashing. Processes that are being executed spend much time on the disk in attempts to be paged. The degree of multiprogramming cannot be increased because processes need to be suspended to increase CPU utilization. (∆ paging is not helping here)

(CPU kept busy)
∴ No. of process ↑

(b) CPU utilization = 87% disk utilization = 3% → paging is helping

CPU at 87% and disk at 3% means that CPU is going to cause the Bottleneck eventually. Disk is being underused and increasing the degree of multiprogramming would inevitable cause thrashing.

(c) CPU utilization 13%, disk utilization 3%.

CPU and disk space both are very low. The obvious answer would be an increase in degree of multiprogramming would increase CPU utilization.

As number of processes are less, paging is not having much effect.

Ans: (a) Thrashing is occuring

(b) CPU utilization is sufficiently high to leave things alone, and increase degree of multiprogramming.

(c) Increase the degree of multiprogramming

SUBMITTED BY:
BHAGYA VINOD RANA
U19CS012
CSE (3rd yr)

vision