Memory - Management Techniques

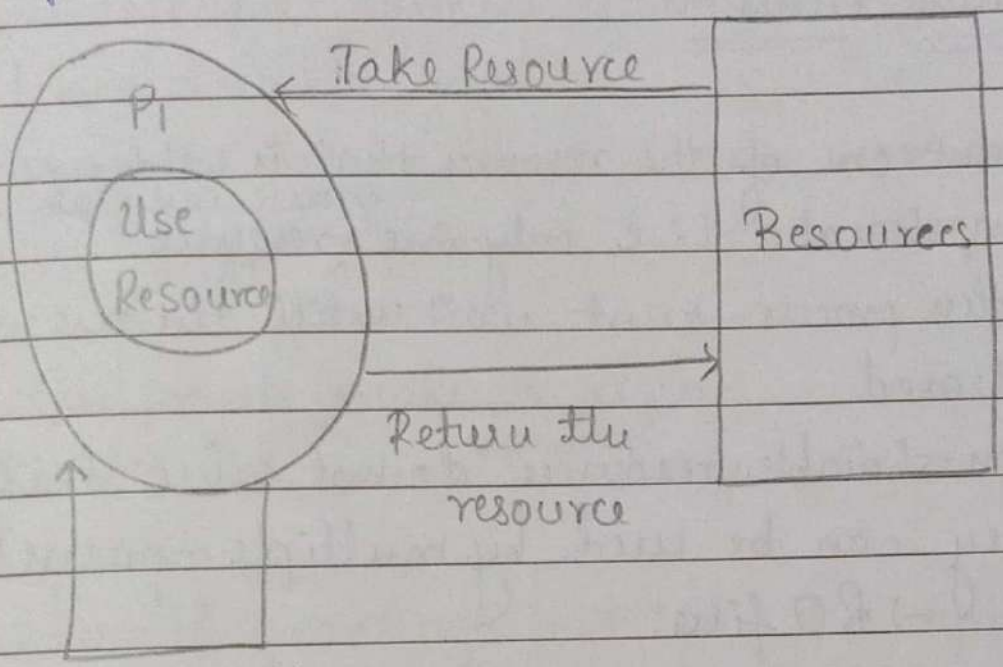
- Critical factor in choosing the technique is:- hardware design
- page table is also main memory.
- demerit : two memory references are required.
- TLB (translation lookaside buffer)
- Program and page table in the main memory.
- Some part of page table (recently used page number and frame number) is in TLB which is faster
- One TLB and one more fetch (TLB match) else TLB miss

- One TLB miss, page table acces, fetch
  - more time in case of TLB miss

E- TLB accus time.

DATE : / /

System Model



A set of processes is in a deadlocked state iff when every process in the set is waiting for an event that can be caused only by other process in the set.

Resources → physical (CPU cycles, memory, printer)
→ logical (files, semaphores, monitors)

Multithreaded programs are good candidates of deadlock ∵ multiple threads can compete for shared resource

Necessary condition for deadlock:

1) Mutual exclusion

→ Atleast one of the resource that is held must be non-sharabe, (i.e only one process can make use of) resource
other process must wait untill the resource is released
Non-sharable resources do not cause deadlock (they can be used by multiple process)
→ RO files.

2) Hold and wait: Process must hold atleast one resource and request for the resources that are held by other process.

3) No preemptoin: Resources cant be preempted i.e processes can voluntarily give up the resources it holds once the task is completed.

4) Circular wait $\{P_0, P_1 - - - P_n\}$

$P_0 \rightarrow P_1$, $P_1 \rightarrow P_2$, $P_2 \rightarrow P_3$, $P_{n-1} \rightarrow P_n$, $P_n \rightarrow P_0$

→ Implies hold and wait

Hence all 4 conditions are not completely independent.

Banker's Algorithm (Overview)

- Suitable for systems with multiple instances of same resource type
- Any new process must declare maximum no. of resource types required.

When process makes a request

- check if the system is in safe state - apply safety algorithm
- chick if granting will leave the system in a safe state - apply resource request algorithm
- If not, process should wait

Bankers Algorithm - Data Structures

n - processes

m - resources

Available : An array of length m. Indicates the no. of available resources of each type

Resource - Allocation Graph

RS:- A B C D A B E A B C D E

(i) FIFO

+	+	+	+	+	−	−	+	+	A	B	+
A	A	A	D	D	D	E					E
	B	B	B	A	A	A					C
		C	C	C	B	B					B

+	E
E	
C	
D	

page fault rate = $\frac{9}{12}$

Optimal

+	+	+	+	A	B	+	A	B	+	+
A	A	A	A			A			C	C
	B	B	B			B			B	D
		C	D			E			E	E

page fault rate = $\frac{7}{12}$

9c

	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	0	0	0	0	0	0
P1	2	0	0	2	0	2			
P2	3	0	3	0	0	0			
P3	2	1	1	1	0	0			
P4	0	0	2	0	0	2			

Sol'n: Finish

~~F~~T	F	~~F~~T	~~F~~T	F
0	1	2	3	4

Work = Available =

A	B	C
0	0	0

Safe sequence exists
<P0, P2, P3, P4, P1>
So no-deadlock.

```
  0 0 0
+ 0 1 0   P0
  0 1 0
  3 0 3   P2
  3 1 3
  2 1 1   P3
  5 2 4
+ 0 0 2   P4
  5 2 6
+ 2 0 0   P1
  7 2 6
```

(4)

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	2	0	0	1	3	4	1	2	1	3	2	2
P1	0	2	0	1	1	4	2	3				
P2	1	1	1	1	2	3	4	2				
P3	1	1	1	1	2	5	5	3				
P4	1	0	1	1	3	5	3	1				
P4'	1	1	1	1								

(i) Safe state?

Need = Max - Allocation

	A	B	C	D
✓ P0	1	4	1	1
✓ P1	1	2	2	2
✓ P2	1	2	3	1
✓ P3	1	4	4	2
✓ P4	2	5	2	0
	2	4	2	0

Initialize

Work = A B C D

Available = 1 3 2 2

Finish =

0	1	2	3
F	F	F	F

Safe sequence = <P1, P0, P4, P2, P3>

Allocation

	A	B	C	D
	1	3	2	2
+ P1	0	2	0	1
	1	5	2	3
+ P0	2	0	0	1
	3	5	2	4
+ P4	1	0	1	1
	4	5	3	5

	A	B	C	D
	4	5	3	5
+ P2	1	1	1	1
	5	6	4	6
+ P3	1	1	1	1
	6	7	5	7

Yes the system has a safe-sequence
$\langle P1, P0, P4, P2, P3 \rangle$

(b) Request from P4 $\langle 0\ 1\ 0\ 0 \rangle$

(i) (Request $\leq$ Need)
$(0,1,0,0) \leq (2,5,2,0)$
yes

(ii) (Request $\leq$ available)
$(0,1,0,0) \leq (1,3,2,2)$
yes

(iii) Allocation
$(1,2,2,2)$

Safe-sequence
$\langle P1, P0, P4, P2, P3 \rangle$

	A	B	C	D
	1	2	2	2
+ P1	0	2	0	1
	1	4	2	3
+ P0	2	0	0	1
	3	4	2	4
+ P4	1	1	1	1
	4	5	3	5
+ P2	1	1	1	1

	5	6	4	6
+ P3	1	1	1	1
	6	7	5	7

Request will leave the system in safe-state so that request can be granted.

PAGE NO.:

DATE: / /

1a.

	Allocation			Request			Available		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P0	1	0	1	0	1	1	0	0	1
P1	1	1	0	1	0	0			
P2	0	1	0	0	0	1			
P3	0	1	0	0	2	0			

Deadlock - detection -OR- safety algorithm.

Work = Available = $\langle$ R1, R2, R3 $\rangle$ = $\langle$ 0, 0, 1 $\rangle$

Finish =

0	1	2	3
~~F~~T	~~F~~T	~~F~~T	~~F~~T

$\langle P2, P0, P1, P3 \rangle$ is safe-state

```
    0  0  1
P2+ 0  1  0
    ------
    0  1  1
P0+ 1  0  1
    ------
    1  1  2
P1+ 1  1  0
    ------
    2  2  2
P3+ 0  1  0
    ------
    2  3  2
```

(3) T1 - 5

T2 - 4.

	Allocation		Max		Available		Need	
	T1	T2	T1	T2	T1	T2	T1	T2
P1	1	1	2	3	1	1	1	2
P2	1	1	3	2			2	1
P3	2	1	4	4			2	3

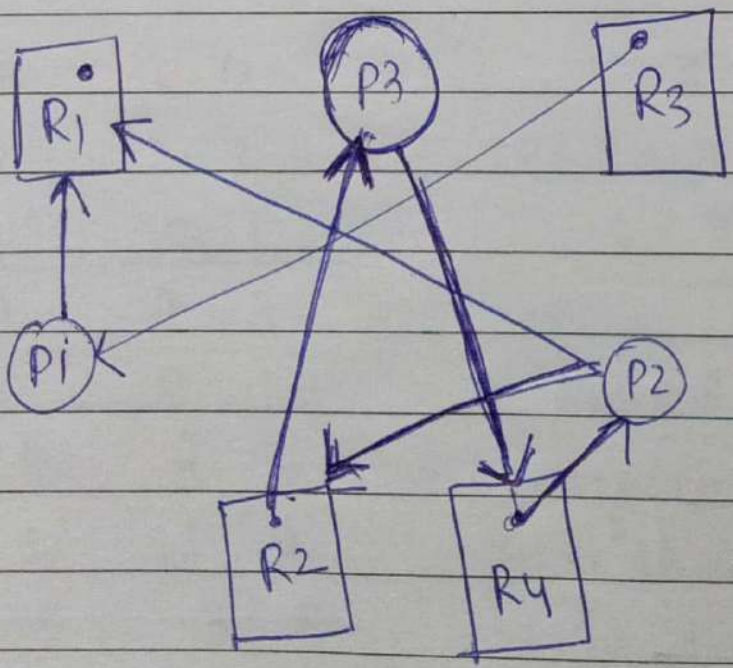
Finish = | F | F | f |

1 2 3

No need can be satisfied from available

So deadlock

(2)



Cycle

P2 → R2 →

P3 → R4 → P2

10K 2K ✓
15K 7K 30
5K 27K ↑ 23K
32K 1K
21K

(i) first fit

2K is put in 10K
8K, 15K, 5K, 32K, 2K.

7K is put in 8K
1K, 15K, 5K, 32K, 2K

27K is put in 32K
1K, 15K, 5K, 5K, 2K

1K is put in 1K
15K, 5K, 5K, 2K.

(ii) Next fit

2K in 10K
8K, 15K, 5K, 32K, 2K
↑

7K in 8K
1K, 15K, 5K, 32K, 2K
↑

27K is put in 32K
1K, 15K, 5K, 5K, 2K
↑

(iii) Best-fit

Order: 2K, 5K, 10K, 15K, 32K

~~Smallest hole is put in~~

2K in 2K
5K, 10K, 15K, 32K

7K in 10K
3K, 5K, 15K, 32K

~~27K in 32K~~
3K, 5K, 5K, 15K

1K is put in 2K
1K, 15K, 5K, 5K, 1K

1K is put in 3K
2K, 5K, 5K, 15K

DATE: / /

2a) Process: 300K, 25K, 12K, 50K

Block: 150K, 350K

(first fit)	(best fit)	(worst)
300K in 350K.	300K in 350K.	300K in 350K
150K, 50K.	50K, 150K	150K, 50K
25K in 150K.	25K in 50K	25K in 150K
125K, 50K	25K, 150K	125K, 50K
125K in 125K	125K in 150K.	125K in 125K
50K in 50K	25K, 25K	50K in 50K

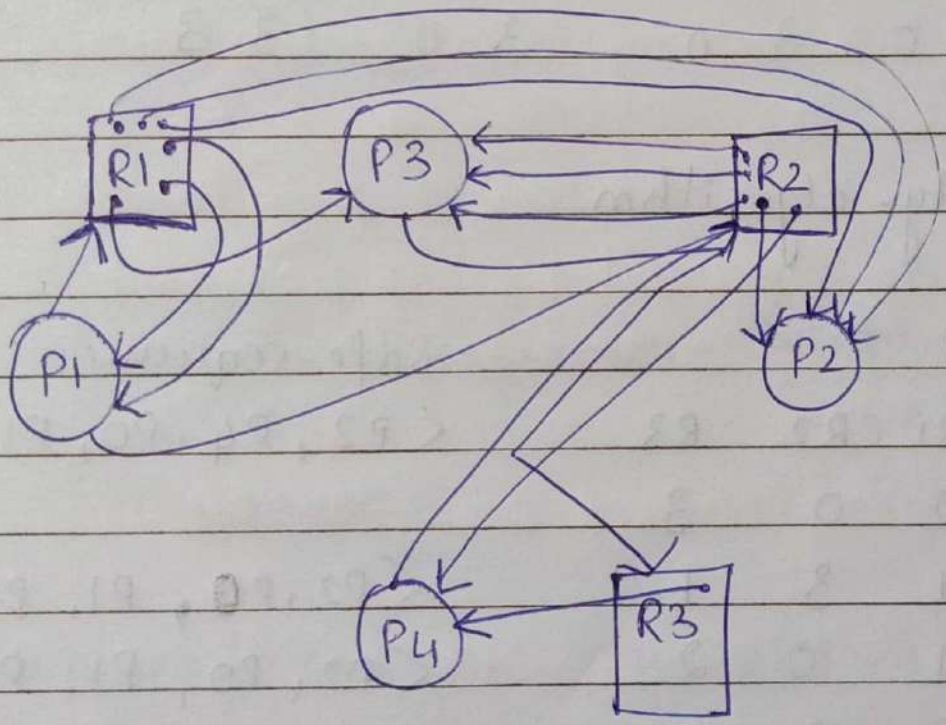
↓

together but not contiguous

DATE : / /

5c

	Allocation			Request			Available		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	2	0	0	1	1	0	0	0	0
P2	3	1	0	0	0	0			
P3	1	3	0	0	0	1			
P4	0	1	1	0	1	0			



6b Steps for page fault

effective memory access time =
hit ratio (memory access time + TLB access
+ (1 - hit ratio) (m + u + TLB)

effective memory acess time = (300ns

EAT = (1-p) x ma + page fault overhead + swap page out + swap page in + restart overhead

(1-p) x 300ns + p x 8000 000

p=0. (no page fault)
EAT = 300ns.