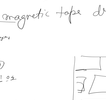
Operating Systems 3

Memory Management - cont.

- Hardly directly
- indirect / binary mapping
- TLB replacement → on context switching
- contiguous memory allocation
  - fixed partitioning
  - fixed & fixed
- Paging → RAM → mapped / directly mapping
  - Page → Frame
- efficiently mapping → binary page
  - binary page → Frame → RAM
  - Page → RAM → HDD
- page replacement

Recap

Memory Hierarchy



Primary → RAM ↔ HDD ↔ Secondary → remote / external storage

Tertiary → Tape, Memory card, Optical disk

→ long term data storage

magnetic tape drives → v.v. cheap

Page Replacement Algorithm

- FCFS → Fairness
- Optimal → Hindsight
- LRU → Least Recently used
- FIFO → Queue
- LIFO → Stack
- MRU → Most recently used
- Optimal (Belady's algorithm)
  - Look ahead
  - Look back

Page Segment

- Segment
  - Big segment → you need to assign
  - Small segment → no waste
  - Optimal segment → efficient
- Segment fragment

3 2 1 0 2 4 3 2 1 0 3

→ RAM → 4 frame / expect 1st to reduce

FIFO suffers from Belady's anomaly

there can be some [additional] cost (cost)

in disk Trade → Page fault → anomaly

expectation Trade → Page fault → replacement

7	2	3	2	2	4	2	4	7	3	3	0
	7	2	3	3	2	4	2	4	7	7	3
		7	2	2	3	2	4	2	4	4	7
			7	7	2	3	2	4	2	2	4
					7	7	3	2	3	3	2
3	2	1	0	2	4	3	2	1	0	3	

RAM → 10 Page faults ?

7 2 3 1 2 4 2 3 2 5

LRU / optimal / stack based algo
↳ do not suffer from Belady's anomaly

7	2	3	0	0	4	4	7	1	1	1	RAM
	2	2	2	2	2	2	2	2	0	0	⇒
		1	1	1	1	3	3	3	3	4	
3	2	1	0	2	4	3	2	1	0	3	Page

consider $\begin{cases} 3 \to 2 \\ 2 \to 3 \\ 1 \to 2 \\ 0 \to 2 \\ 4 \to 3 \end{cases}$ 10 faults

7	2	3	0	0	4	4	7	1	1	1	RAM
	2	2	2	2	2	2	2	2	0	0	⇒
		1	1	1	1	3	3	3	3	4	
3	2	1	0	2	4	3	2	1	0	3	Page

10 faults

LRU

can not suffer from Belady's anomaly

WINDOW → LRU → approximate clock algo

Linux → 2.2

LRU variant

optimal Algo

↳ Cannot be implemented → future algo
↳ Hindsight

↳ useful theoretically

↳ " for reference tests

7	2	3	0	0	4	4	7	1	1	1	RAM
	2	2	2	2	2	2	2	2	0	0	⇒
		1	1	1	1	3	3	3	3	4	
3	2	1	0	2	4	3	2	1	0	3	Page

7 faults

thrashing → old → 60's

Cache dog → indirectly boosts up

Threading & Synchronization

Race Condition

↳ breaking system

Rough	Sudden	Rasp
100%	100%	200%

void bug - Equation (Cpp, web)

if (bugs.count 2) {int} → goal!

Bug count → 2 → 3 → 4 → 5

3 3

AL = load bugs.count
SUB ax, AL
SBB AL, bugs.count

Critical section

Counter = 2 → RAM 3

Thread 1	Thread 2
Process 1	Process 2
LOAD Counter AL	LOAD Counter AL
INC AL	INC AL
SAVE AL counter	SAVE AL counter

processing buffers of CPU
load in memory

Load
Call
Save

Text
Text
global
Text
Text
Text