

Sistemi Operativi

AA 2018/19

Esercitazione

03 Dicembre 2018

Esercizio 1

Come si può implementare l'algoritmo di sostituzione delle pagine *Second Chance*?

Soluzione L'algoritmo *Second Chance* - o altresì noto come *clock algorithm* - è un algoritmo FIFO per la sostituzione delle pagine che usa un reference bit (implementato in hardware) per capire quale pagina bisogna rimpiazzare. In particolare:

- se il reference bit di una pagina è 0 allora essa viene rimpiazzata;
- se il reference bit è 1 il bit viene settato a 0 e la pagina viene lasciata in memoria, quindi si passa alla pagina successiva - usando le stesse regole.

Un modo per implementare tale algoritmo è attraverso una *coda circolare* con un puntatore alla pagina da rimpiazzare - che scorre nella coda finché non trova una pagina con reference bit pari a 0. Trovata una pagina che soddisfa le richieste, viene eliminata dalla coda e rimpiazzata con la nuova. Il funzionamento dell'algoritmo è illustrato in Figura 1.

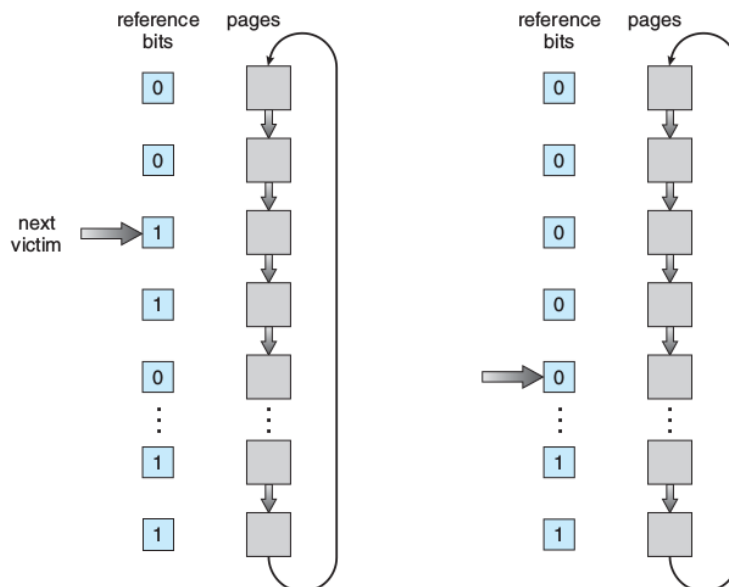


Figure 1: L'algoritmo di sostituzione delle pagine *Second Chance*. Il puntatore scorre finché una pagina con reference bit 0 non viene trovata; quindi la nuova pagina prenderà il posto di quest'ultima nella coda circolare.

Esercizio 2

Sia data la seguente traccia di accesso alle pagine di memoria:
 7 - 0 - 1 - 2 - 0 - 3 - 0 - 4 - 2 - 3 - 0 - 3 - 2.

Si assuma di avere un TLB di 3 elementi, gestito con politica *Optimal replacement*. Si assuma che un ciclo di fetch duri T_{fetch} ed un accesso al TLB impieghi T_{TLB} .

Domanda Di quanto aumentano le prestazioni con un TLB a 4 elementi?

Soluzione Le tracce di accesso sono riportate rispettivamente in Figura 2a e Figura 2b.

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

(a) Traccia con TLB di 3 elementi.

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

(b) Traccia con TLB di 4 elementi.

Figure 2: Tracce d'accesso con TLB a 3 e 4 elementi. In rosso sono evidenziate le sostituzioni, in verde gli *hit*.

Nel caso di TLB a 3 elementi il numero di hit e' pari a $N_{hit} = 6$, con una percentuale di *page fault* del 54%; passando a 4 elementi si avra' $N_{hit} = 7$ ed una percentuale di *page fault* del 46%. Il guadagno prestazionale e' quindi pari a circa l'8%.

Esercizio 3

Sia data la seguente traccia di accesso alle pagine di memoria:

8 9 1 1 4 8 1 1 1 2 3 7 5 2 3.

Domanda Si assuma di avere un TLB di 4 slot gestito in modalita' *LRU*. Di quanto migliorano le prestazioni della memoria raddoppiandolo?

Soluzione La tracce di accesso nei due casi sono raffigurate rispettivamente nelle Figure 3a e 3b.

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

(a) Traccia con TLB a 4 slot

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

(b) Traccia con TLB a 8 slot

Figure 3: Illustrazione della traccia di accesso alle pagine di memoria. In verde indichiamo un *hit* nel TLB, mentre in rosso viene indicata la pagina candidata ad eliminazione - secondo la politica LRU.

Quindi, supponendo che il TLB abbia 4 unita', il tempo di accesso medio T_{mean}^4 e' dato dalla seguente relazione:

$$T_{mean}^4 = 7T_{hit} + 8T_{miss}$$

dove

$$\begin{aligned} T_{hit} &= T_{TLB} + T_{fetch} \\ T_{miss} &= 2T_{TLB} + 2T_{fetch} \end{aligned}$$

Analogamente, nel caso in cui il TLB abbia 8 unita', il tempo medio sara' calcolato come segue:

$$T_{mean}^8 = 7T_{hit} + 8T_{miss}$$

E' facilmente intuibile da un semplice confronto che in questo particolare caso aumentare la dimensione del TLB non influenza le prestazioni.

Esercizio 4

Sia dato un sistema con paginazione con frame di dimensione 200 e TLB a 3 entries, gestito con politica di rimozione della pagine *LRU*. Si ipotizzi che un piccolo processo sia completamente contenuto nella prima pagina del sistema - i.e. da locazione 0 a 199. Si consideri quindi un array bidimensionale `int A[] []` con dimensioni 100×100 . Calcolare il numero di *page fault* dovute all'inizializzazione dell'array nelle seguenti modalita':

Modalita' 1

```

1  for (int r = 0; r < 100; ++r)
2      for (int c = 0; c < 100; ++c)
3          A[r][c] = 0;
4
5

```

Modalita' 2

```

1  for (int c = 0; c < 100; ++c)
2      for (int r = 0; r < 100; ++r)
3          A[r][c] = 0;
4
5

```

Soluzione Le traccia di accesso alle pagine nel primo e nel secondo caso sono illustrate rispettivamente nella Figura 4a e Figura 4b.

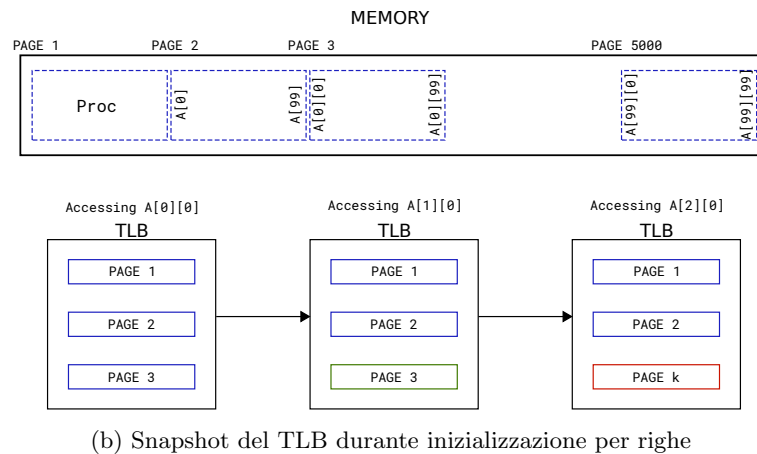
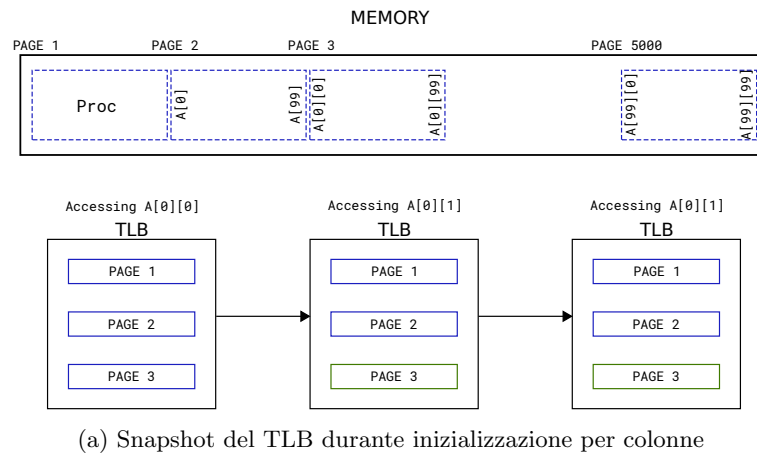


Figure 4: Illustrazione della traccia di accesso alle pagine in memoria. Si noti che inizializzando per *righe* - i.e. modalita' 2 - richiede uno swap di pagina ogni due accessi.

Come illustrato in Figura 4, nella modalita' 1 accederemo alle pagine in maniera sequenziale - per come e' allocata la memoria. Nella modalita' 2, l'accesso alla memoria sara' scattered, forzando a swappare pagina ogni 2 accessi. Da qui avremo che il numero di *page fault* sara'

$$\star \text{ Modalita' 1} \rightarrow N_{fault} = 50$$

$$\star \text{ Modalita' 2} \rightarrow N_{fault} = 5000$$

Esercizio 5

Siano dati due processi P1 e P2. Si supponga che la loro traccia di accesso alle pagine in memoria sia quella illustrata nella Figura 5. Si supponga che ogni pagina abbia dimensione 4MB, che la memoria fisica disponibile sia 8MB e che sia presente un disco fisico meccanico da 16 GB dedicato allo swap. Cosa succede nell'istante di tempo $t = 5$?

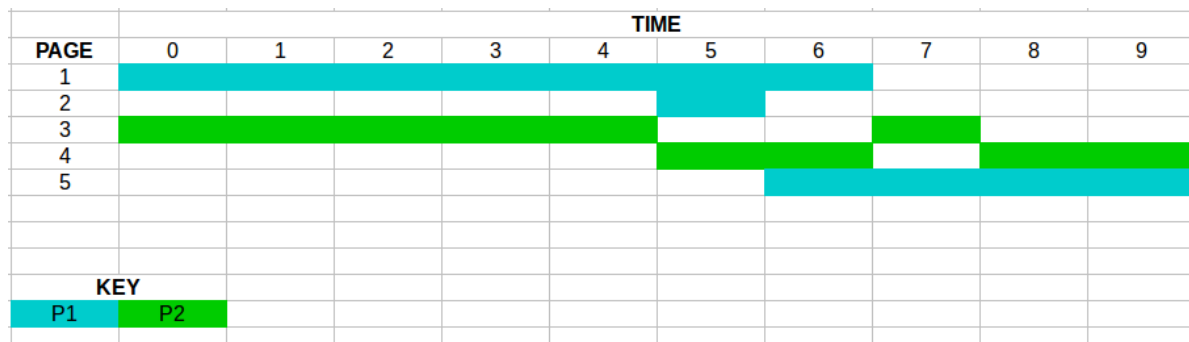


Figure 5: Traccia di accesso alle pagine dei processi P1 e P2 - rispettivamente in ciano e verde.

Soluzione In questo caso, il nostro sistema potrà contenere al massimo 2 pagine di memoria in RAM, ergo $D_{RAM} = 2$, mentre i *working set* dei due processi avranno dimensione $WSS_1 = 2$ e $WSS_2 = 1$. Ciò implica che - a partire dal tempo $t = 5$ fino a $t = 6$ - si verificherà il fenomeno del *thrashing*, ovvero il processo impegna più tempo nello *swapping-in/out* delle pagine che nell'esecuzione del processo stesso. Ciò poiché il *working set* dei due processi non può essere contenuto in RAM, in formulæ:

$$\sum_p WSS_p > D_{RAM}$$

Per evitare tale fenomeno, bisognerebbe aumentare la dimensione della RAM almeno fino a 12MB - lo spazio minimo necessario a contenere 3 pagine da 4MB - oppure killare un processo.