Sistemi Operativi 1(solo esercizi)

1-Esercizi sulle memorie

1.1-Allocazione di processi in blocchi

Dato un processo P che necessita di una memoria libera di N KiB per essere allocato in modo contiguo, avendo blocchi liberi $B_1, B_2, ..., B_n$ di dimensione $M_1, M_2, ..., M_n$ il processo verrà allocato in blocchi diversi in base alla politica:

- Best-Fit: blocco più piccolo tale che $M_i > N$
- Worst-Fit: blocco più grande tale che $M_i > N$
- First-Fit: primo blocco tale che $M_i > N$

Esempio:

Si supponga che un processo P necessiti di un'area di memoria libera pari a 115 KiB per essere allocato in modo contiguo in memoria principale. Se la lista dei blocchi di memoria libera contiene i seguenti elementi:

A	B	C	D	E	F
300 KiB	600 KiB	350 KiB	200 KiB	750 KiB	125 KiB

In base alla politica P verrà allocato:

- Best-Fit: F perché è il blocco minore in cui $M_F > 115\ \text{KiB}$
- Worst-Fit: E perché è il blocco maggiore in cui $M_E > 115\ \text{KiB}$
- First-Fit: A perché è il primo blocco in cui $M_A > 115\ \text{KiB}$

1.2-Memoria divisa in pagine

Data una memoria fisica di grandezza M con indirizzamento con word size w e la memoria sia divisa in pagine di grandeza S, la dimensione della page table, cioè il numero di entry:

$$\text{T} = \frac{M}{S}$$

Invece il numero di bit necessari per indirizzare le word sarà:

$$\text{n}^\circ \text{ bit word} = \log_2(\frac{M}{w})$$

Il numero di bit necessari per l'indice di pagina e per l'offset sono:

$$\text{n}^\circ \text{ bit offset} = \log_2(\frac{S}{w})$$
$$\text{n}^\circ \text{ bit indice pagina} = \log_2(\frac{M}{S})$$

Esempio:

Si supponga di avere una memoria M di capacità pari a $8\ \text{KiB} = 8.192\ \text{B}$. Assumendo che l'indirizzamento avvenga con lunghezza di parola (word size) pari al singolo byte e che M utilizzi una gestione paginata con blocchi di dimensione pari a $S = 128\ \text{B}$, quale dimensione (intesa come numero di entry) ha la corrispondente page table T?

$\text{T} = \frac{8192}{128} = 64$

1.3-Pagina e offset di un indirizzo

Data una memoria fisica di grandezza M con frame di grandezza F, l'indirizzo A avrà numero di pagina p e offset:

$$p = \lfloor \frac{A}{F} \rfloor$$
$$\text{offset} = A \% F$$

Esempio:

Si consideri una memoria M di capacità pari a $100\ \mathrm{B}$ con frame di dimensione pari a $10\ \mathrm{B}$. Dato l'indirizzo del byte $A = 37$, quale sarà l'indirizzo di pagina e l'offset?

$p = \lfloor \frac{37}{10} \rfloor = 3$

$\text{offset} = 37\%10 = 7$

1.4-Controllare i page fault

Data una memoria con k frame e un processo con n pagine virtuali per calcolare il numero di page fault bisogna prima controllare quali sono le pagine caricate in memoria(al massimo k insieme) e vedere se la pagina richiesta in quel momento è caricata in memoria, se non lo è si verifica un page fault.

Quando una pagina viene richiesta e non è già caricata, se la memoria non è piena viene caricata senza sovrascriverne nessun'altra.

Nel caso in cui la memoria è piena quale pagina verrebbe sovrascritta dipende dal protocollo:

- First In First Out(FIFO): viene sovrascritta la pagina che è caricata in memoria da più tempo
- Last Recent Used(LRU): viene sovrascritta la pagina che non viene chiamata da più tempo
- MIN(OPT): viene sovrascritta la pagina che non verrà acceduta per il lasso di tempo maggiore nel futuro

Esempio:

Data una memoria composta da 3 frame fisici e un processo composto da 5 pagine virtuali: A, B, C, D, E, si calcoli il numero di page fault che si verificano a fronte delle seguenti richieste da parte del processo: B, C, C, B, A, E, B, A, E, D, B. Si assuma che nessuna pagina del processo sia inizialmente caricata in memoria e che si utilizzi un algoritmo LRU di sostituzione delle pagine.

Pagina chiamata	Pagine	in	memoria	Page fault
B	/	/	/	Si
C	/	/	B	Si
C	/	B	C	No
B	/	B	C	No
A	/	C	B	Si
E	C	B	A	Si
B	B	A	E	No
A	A	E	B	No
E	E	B	A	No
D	B	A	E	Si
B	A	E	D	Si

Il numero di page fault è 6.

2-Calcolo di tempi

2.1-Tempo di accesso in memoria

Data una memoria fisica con tempo di accesso in memoria t_{MA}, tempo di gestione di un page fault t_{FAULT} che accade con probabilità p_{FAULT}, il tempo di accesso medio sarà:

$$t_A = (1-p)t_{\text{MA}} + p \cdot t_{\text{FAULT}}$$

> La probabilità è scritta come un numero $p = [0, 1]$, per convertirlo in percentuale bisogna moltiplicarlo per 100.

2.2-Tempo medio di waiting

Data una coda di n processi, ognuno con un istante di arrivo $\mathrm{T}_{\text{arrival}}$ e un tempo di completamento CPU burst, il tempo di waiting di un determinato processo i si calcola:

$$\mathrm{T}^{i}_{\text{waiting}} = \underbrace{\mathrm{T}_{\text{completion}}}_{\substack{\text{istante in} \\ \text{cui termina}}} - \mathrm{T}^{i}_{\text{arrival}} - \text{CPU burst}^{i}$$

Da cui il tempo medio di waiting dell'intera coda di n processi sarà:

$$\overline{\mathrm{T}}_{\text{waiting}} = \frac{1}{n}\sum_{i=0}^{n}\mathrm{T}^{i}_{\text{waiting}}$$

Per calcolare l'istante di completamento di un processo bisogna controllare la coda e cambia in base al protocollo usato:

- First Come First Served(FCFS): viene eseguito per intero il processo con $\mathrm{T}_{\text{arrival}}$ minore e poi viene eseguito il successivo
- Round Robin(RR): si inizia l'esecuzione dei processi in ordine secondo la cosa ma fermandoli e passando al successivo ogni tot di tempo, detto time slice q. Ogni volta che un processo viene fermato viene aggiunto alla fine della coda.
- Shortest Job First(SJF): si esegue il processo con il CPU burst minore e ci sono due tipi possibili:
  - Non-Preemptive: una volta che un processo è iniziato viene portato a termine
  - Preemptive: ogni volta che un nuovo processo arriva, se ha CPU burst minore rispetto a quello rimanente al processo in corso, viene eseguito questo nuovo processo

  Nel caso in cui ci siano delle attività I/O di durata $\mathrm{T}_{\mathrm{I/O}}$ il tempo di waiting di quel determinato processo diventa:

$$\mathrm{T}^{i}_{\text{waiting}} = \underbrace{\mathrm{T}_{\text{completion}}}_{\substack{\text{istante in} \\ \text{cui termina}}} - \mathrm{T}^{i}_{\text{arrival}} - \text{CPU burst}^{i} - \mathrm{T}_{\mathrm{I/O}}$$

Esempio 1:

Data una coda di processi:

Job	$T_{arrival}$	T_{burst}
A	0	3
B	2	7
C	6	4
D	7	5

1. FCFS:
   L'ordine dei processi sarà:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A			B							C				D				

   Il tempo di waiting sarà:

$$T^{A}_{waiting} = 3 - 0 - 3 = 0$$
$$T^{B}_{waiting} = 10 - 2 - 7 = 1$$
$$T^{C}_{waiting} = 14 - 6 - 4 = 4$$
$$T^{D}_{waiting} = 19 - 7 - 5 = 7$$

$$\overline{T}_{waiting} = \frac{0 + 1 + 4 + 7}{4} = 3$$

2. RR con $q = 4$:
   L'ordine dei processi sarà:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A			B				C				D				B			D

   Il tempo di waiting sarà:

$$T^{A}_{waiting} = 3 - 0 - 3 = 0$$
$$T^{B}_{waiting} = 18 - 2 - 7 = 9$$
$$T^{C}_{waiting} = 11 - 6 - 4 = 1$$
$$T^{D}_{waiting} = 19 - 7 - 5 = 7$$

$$\overline{T}_{waiting} = \frac{0 + 9 + 1 + 7}{4} = 4.25$$

Esempio 2:

Data una coda di processi:

Job	$\mathrm{T}_{\text{arrival}}$	$\mathrm{T}_{\text{burst}}$
A	0	6
B	1	3
C	2	7
D	3	4

1. SJF Non-Preemptive:
   L'ordine dei processi sarà:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A						B			D				C						

   Il tempo di waiting sarà:

$$\mathrm{T}^{A}_{\text{waiting}} = 6 - 0 - 6 = 0$$
$$\mathrm{T}^{B}_{\text{waiting}} = 9 - 1 - 3 = 5$$
$$\mathrm{T}^{C}_{\text{waiting}} = 20 - 2 - 7 = 11$$
$$\mathrm{T}^{D}_{\text{waiting}} = 13 - 3 - 4 = 6$$

$$\overline{\mathrm{T}}_{\text{waiting}} = \frac{0 + 5 + 11 + 6}{4} = 5.5$$

2. SJF Preemptive:
   L'ordine dei processi sarà:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	B			D				A					C						

   Il tempo di waiting sarà:

$$\mathrm{T}^{A}_{\text{waiting}} = 13 - 0 - 6 = 7$$
$$\mathrm{T}^{B}_{\text{waiting}} = 4 - 1 - 3 = 0$$
$$\mathrm{T}^{C}_{\text{waiting}} = 20 - 2 - 7 = 11$$
$$\mathrm{T}^{D}_{\text{waiting}} = 8 - 3 - 4 = 1$$

$$\overline{\mathrm{T}}_{\text{waiting}} = \frac{7 + 0 + 11 + 1}{4} = 4.75$$

3-Dischi magnetici

3.1-Tempo di trasferimento dati

Dato un disco magnetico in cui si devono trasferire dei dati, con tempo di posizionamento Seek, delay di rotazione ROT e tempo di trasferimento TT, il tempo totale di trasferimento dei dati sarà:

$$\mathrm{T} = \mathrm{Seek} + \mathrm{ROT} + \mathrm{TT}$$

In cui TT dipende dalla velocità di trasferimento TR e dalla quantità di dati trasferiti D:

$$\mathrm{TT} = \frac{\mathrm{D}}{\mathrm{TR}}$$

Esempio:

$\mathrm{T} = 40\ \mathrm{ms}$

$\mathrm{Seek} = 18\ \mathrm{ms}$

$\mathrm{ROT} = 7\ \mathrm{ms}$

$\mathrm{TR} = 5\ \mathrm{Gbit/s}$

$40\ \mathrm{ms} = 18\ \mathrm{ms} + 7\ \mathrm{ms} + \frac{\mathrm{D}}{5\ \mathrm{Gbit/s}} \implies \mathrm{D} = 15\ \mathrm{ms} \cdot 5\ \mathrm{Gbit/s} = 15\ \mathrm{ms} \cdot 5\ \mathrm{Mbit/ms} = 75\ \mathrm{MB} = 9.375\ \mathrm{MB}$

3.2-Scheduling del disco

Dato un disco con k tracce(numerate da 0 a k) la cui testina è inizialmente posizionata sulla traccia x_0 ed effettuiamo delle richieste $x_1, x_2, ..., x_n$ la testina percorrerà un certo numero di tracce per soddisfarle tutte che dipendono dall'algoritmo:

- First Come First Served(FCFS): la testina si muove semplicemente in ordine in base alle richieste
- Shortest Seek Time First(SSTF): la testina ogni volte che si deve muovere lo fa verso la richiesta più vicina alla posizione in cui sta
- SCAN: la testina si muove da un'estremo all'altro(da 0 a k e poi torna indietro) del disco e esegue le richieste durante il passaggio
- LOOK: la testina si muova da un'estremo all'altro ma cambiando direzione quando raggiunge la richiesta più esterna
- C-SCAN: la testina si muove da un'estremo all'altra ma leggendo solo solo in una direzione(da k a 0 e poi torna indietro senza leggere per poi rileggere solo durante il tragitto tra k e 0)
- C-LOOK: la testina si muove da un'estremo all'altra ma leggendo solo solo in una direzione ma cambiando direzione quando raggiunge la richiesta più esterna

Il numero di tracce percorse è la somma dei singoli spostamenti.

Esempio:

Avendo un disco con 100 tracce effettuiamo le richieste 65, 40, 18, 78 e la testina è inizialmente posizionata sulla traccia 30(i numeri sulle frecce indicano il numero di tracce percorse durante quello spostamento):

- FCFS: $30 \xrightarrow{35} 65 \xrightarrow{25} 40 \xrightarrow{22} 18 \xrightarrow{60} 78 \implies \text{n° passi} = 35 + 25 + 22 + 60 = 142$
- SSTF: $30 \xrightarrow{10} 40 \xrightarrow{22} 18 \xrightarrow{47} 65 \xrightarrow{13} 78 \implies \text{n° passi} = 10 + 22 + 47 + 13 = 92$
- SCAN: $30 \xrightarrow{12} 18 \xrightarrow{18} 0 \xrightarrow{40} 40 \xrightarrow{25} 65 \xrightarrow{13} 78 \implies \text{n° passi} = 12 + 18 + 40 + 25 + 13 = 108$
- LOOK: $30 \xrightarrow{12} 18 \xrightarrow{22} 40 \xrightarrow{25} 65 \xrightarrow{13} 78 \implies \text{n° passi} = 12 + 22 + 25 + 13 = 72$
- C-SCAN: $30 \xrightarrow{12} 18 \xrightarrow{18} 0 \xrightarrow{100} 100 \xrightarrow{22} 78 \xrightarrow{13} 65 \xrightarrow{25} 40 \implies \text{n° passi} = 12 + 18 + 100 + 22 + 13 + 25 = 190$
- C-LOOK: $30 \xrightarrow{12} 18 \xrightarrow{60} 78 \xrightarrow{13} 65 \xrightarrow{25} 40 \implies \text{n° passi} = 12 + 60 + 13 + 25 = 110$

Cose utili varie

- Quando si esegue un `fork()` il pid del figlio è uguale a 0.