Sistemi Operativi
23 giugno 2011
Compitino II B

Si risponda ai seguenti quesiti, giustificando le risposte.

1. Si descriva il fenomeno del *thrashing* e si indichino tecniche per ovviare a tale fenomeno.

   Risposta: (4 punti) Quando la memoria fisica libera (e quindi il numero di frame liberi) è insufficiente a contenere il working set corrente di un processo, quest'ultimo comincerà presto a generare parecchi page fault, rallentando considerevolmente la propria velocità d'esecuzione. Quando parecchi processi cominciano ad andare in thrashing, ovvero a spendere più tempo per la paginazione che per l'esecuzione, il sistema operativo potrebbe erroneamente essere indotto a dedurre che sia necessario aumentare il grado di multiprogrammazione (dato che la CPU rimane per la maggior parte del tempo inattiva a causa dell'intensa attività di I/O). In questo modo vengono avviati nuovi processi che però, a causa della mancanza di frame liberi, cominceranno a loro volta ad andare in thrashing: in breve le prestazioni del sistema collassano fino ad indurre l'operatore a dover terminare forzatamente alcuni processi.

   Per ovviare a tale fenomeno si può aumentare la memoria fisica oppure cercare di approssimare correttamente le località dei processi controllandone il working set ed allocando loro frame sufficienti per coprirlo. Quindi, alla creazione di un nuovo processo, questo viene ammesso nella coda ready solo se ci sono frame liberi sufficienti per coprire il suo working set. Altrimenti si sospende uno dei processi per liberare la sua memoria per il nuovo processo (lo scheduling di medio termine diminuisce il grado di multiprogrammazione). In questo modo si impedisce il thrashing, massimizzando nel contempo luso della CPU.

2. (a) Si descriva il concetto di *working set* $WS(t, \Delta)$, all'istante t con intervallo Δ.
   (b) Si consideri la seguente stringa di riferimenti (partendo con $t = 0$):

   7 5 6 7 7 6 6 1 1 5 3

   Cosa è $WS(10, 8)$, ossia dopo l'ultimo accesso?
   (c) Nel precedente esempio quanti page fault ci sono complessivamente con $\Delta = 3$ (supponendo che in ogni istante si mantenga in memoria esattamente il solo working set)?

   Risposta:
   (a) (3 punti) Il working set è un'approssimazione della località del processo, ossia è l'insieme di pagine "attualmente" riferite. In generale $WS(t, \Delta)$=insieme delle pagine riferite negli accessi $[(t - \Delta + 1), t]$.
   (b) (2 punti) $WS(10, 8) = \{1, 3, 5, 6, 7\}$.
   (c) (3 punti) Si verificano 6 page fault, come risulta dalla seguente simulazione dell'algoritmo:

7	5	6	7	7	6	6	1	1	5	3
7	5	6	7	7	6	6	1	1	5	3
	7	5	6	6	7	7	6	6	1	5
		7	5							1
p	p	p					p		p	p

3. Si consideri un processo che generi la seguente stringa di riferimenti alle pagine virtuali:

   1 2 0 0 4 2 1 4 1 2

   (a) Se il processo ha 4 frame, gestiti LRU, quanti page fault vengono generati?
   (b) Qual è il numero minimo di frame necessario per minimizzare i page fault?

   Risposta:
   (a) (3 punti) Simuliamo il funzionamento di LRU nel caso della reference string data:

	1	2	0	0	4	2	1	4	1	2
		1	2	2	0	4	2	1	4	1
			1	1	2	0	4	2	2	4
					1	1	0	0	0	0
	P	P	P		P					

Si verificano quindi quattro page fault.

(b) (3 punti) Il minimo numero di page fault è 4 page fault (perché il processo accede a 4 pagine). Per determinare il numero minimo di frame per avere solo 4 page fault, si può sfruttare la *distance string*, che nel caso in questione risulta essere la seguente:

$$\infty \ \infty \ \infty \ 1 \ \infty \ 3 \ 4 \ 3 \ 2 \ 3$$

Si ricorda che la distance string rappresenta la distanza fra la posizione di una pagina nel modello e la prima posizione, ovvero, quella nella prima riga della matrice (contando anche la casella di partenza) nel momento in cui la pagina stessa viene riferita. Se una pagina non è presente nella matrice, allora la sua distanza, quando viene riferita è ∞.

Indichiamo ora con C_i il numero di volte che il numero i compare nella distance string; nel caso in questione abbiamo: $C_1 = 1$, $C_2 = 1$, $C_3 = 3$, $C_4 = 1$, $C_\infty = 4$. Indicando poi con m il numero di frame e con n il numero più grande che compare nella distance string, indichiamo con $F_m = \Sigma_{k=m+1}^{n} C_k + C_\infty$ il numero di page fault che si verificano con m frame e con la reference string data. L'intuizione è la seguente: se ho a disposizione m frame i page fault saranno provocati dai riferimenti a pagine che "distano" almeno $m+1$ dal top della matrice e dal numero di ∞ (ovvero da riferimenti a pagine non ancora presenti nel modello). Nel nostro caso abbiamo: $F_1 = 9$, $F_2 = 8$, $F_3 = 5$, $F_4 = 4$ quindi il numero minimo di frame che minimizza i page fault è 4.

4. Quando un'interruzione viene definita *precisa* (si elenchino le quattro condizioni)?

   Risposta: (4 punti) Un'interruzione si dice precisa quando gode delle seguenti quattro proprietà:

   1. il program counter viene salvato in un posto noto,
   2. tutte le istruzioni che precedono quella puntata dal program counter sono state completamente eseguite,
   3. nessuna delle istruzioni che seguono quella puntata dal program counter è stata eseguita,
   4. lo stato di esecuzione dell'istruzione puntata dal program counter è noto.

5. Si spieghi come funziona l'allocazione indicizzata dei blocchi di un disco. Come viene implementata negli inode in UNIX?

   Risposta: (4 punti) Nel caso dell'allocazione indicizzata si mantengono tutti i puntatori ai blocchi di un file in una tabella indice, memorizzata a sua volta in un blocco su disco (blocco indice). In questo modo per accedere all'i-esimo blocco del file è sufficiente seguire l'i-esimo puntatore del blocco indice. Inizialmente (in fase di creazione del file) tutti gli indici sono inizializzati con un valore (-1) indicante che non vi è nessun blocco dati puntato. In questo modo, per allocare un nuovo blocco al file, è sufficiente assegnare l'indirizzo di un blocco dati disponibile alla prima entry del blocco indice marcata con -1. Così, sacrificando un blocco per la tabella indice, è facile implementare l'accesso random ai file ed evitare il problema della frammentazione esterna.

   In un sistema UNIX l'allocazione indicizzata viene implementata mediante i seguenti metadati presenti negli inode:

   - blocchi diretti: puntatori ai primi 12 blocchi del file,
   - primo indiretto: indirizzo del blocco indice dei primi indiretti,
   - secondo indiretto: indirizzo del blocco indice dei secondi indiretti,

   In base a ciò segue che la dimensione massima di un file, supponendo di disporre di blocchi da 4 KB e puntatori a 32 bit è

$$\begin{aligned} L_{max} &= 12 + 1024 + 1024^2 + 1024^3 \\ &> 1024^3 = 2^{30} blk \\ &= 2^{42} byte = 4\,TB \end{aligned}$$

   dato che con blocchi da 4KB e puntatori a 32 bit (i.e., 4 byte) ogni blocco può contenere 1024 indirizzi.

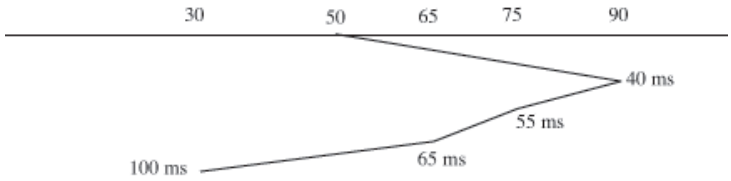
Sistemi Operativi
23 giugno 2011
Compitino II B

6. Si consideri un disco gestito con politica LOOK. Inizialmente la testina è posizionata sul cilindro 50; lo spostamento ad una traccia adiacente richiede 1 ms. Al driver di tale disco arrivano richieste per i cilindri 90, 65, 75, 30, rispettivamente agli istanti 0 ms, 20 ms, 40 ms, 55 ms. Si trascuri il tempo di latenza.

    1. In quale ordine vengono servite le richieste?
    2. Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene effettivamente servita. Qual è il tempo di attesa medio per le quattro richieste in oggetto?

    Risposta:

    1. (3 punti) Le richieste vengono servite nell'ordine 90, 75, 65, 30:



    2. (2 punti) Il tempo di attesa medio per le quattro richieste in oggetto è $\frac{(40-0)+(65-20)+(55-40)+(100-55)}{4} = \frac{40+45+15+45}{4} = \frac{145}{4} = 36,25\ ms.$