SISTEMI OPERATIVI E LABORATORIO
(Esonero e Scritto - Indirizzo Sistemi e Reti)
13 luglio 2007

Cognome:_____Nome:_____
Matricola:_____

Ricordate che non potete usare calcolatrici o materiale didattico. Siate sintetici nelle vostre risposte, anche quando è richiesto di motivarle, sono sufficienti poche righe per rispondere correttamente. (Si ricorda che gli studenti degli anni precedenti devono sostenere l'intero scritto).

Scelgo di svolgere (solo per chi consegna dopo la prima ora):
[ ] la parte relativa alla teoria. **[ ] la parte relativa al laboratorio UNIX**

ESERCIZI RELATIVI ALLA PARTE DI TEORIA DEL CORSO

(il punteggio conseguito farà media con quello ottenuto nella parte di laboratorio. E' comunque necessario prendere almeno 18 punti per considerare passata la parte di teoria.)

ESERCIZIO 1 (9 punti)

Quattro processi arrivano al tempo indicato e consumano la quantità di CPU indicata nella tabella sottostante)

Processo	T. di arrivo	Burst
P1	0	10
P2	1	8
P3	1	6
P4	11	3

a) Calcolare il turnaround medio e il waiting time medio per i processi nel caso dell'algoritmo di scheduling SJF preemtpive (shortest remaining time first). RIPORTANDO IL DIAGRAMMA DI GANTT USATO PER IL CALCOLO.

Turnaround medio:
(0)….P1 …(1) …. P3….(7)….P2….(11)….P4….(14)….P2…(18)….P1….(27)
P1 = 27; P2 = 17; P3 = 6; P4 = 3; 53/4 = 13,25

Waiting time medio:
(basta ricordarsi di sottrarre al turnaround di ogni processo, la durata del suo burst):
P1 = 17; P2 = 9; P3 = 0; P4 = 0; 26/4 = 6,5

b) SJF premptive potrebbe essere usato in un sistema time sharing? E SJF non preemptive? (motivate la vostra risposta)

No, in entrambi i casi, perché non garantiscono assenza di starvation. Infatti, potrebbero arrivare in coda di ready sempre nuovi processi con un CPU time inferiore a processi già in coda e in attesa di essere schedulati.

c) Qual è il più grave problema che può avere un algoritmo di scheduling a priorità, e come si risolve?

La possibilità di starvation (come nel caso della risposta b). Con un meccanismo di aging.

d) Per un sistema time sharing è meglio usare un algoritmo di scheduling preemptive o uno non preemptive? (motivate la vostra risposta)

E' necessario usare un algoritmo preemptive (ma non SJF, ad esempio, RR, che non produce starvation) in modo da garantire che qualsiasi processo di qualsiasi utente possa prima o poi usare la CPU

e) Riportate il diagramma di stato della vita di un processo.

Vedere il lucido della sezione 3.1.2.

ESERCIZIO 2 (9 punti)

In un sistema, la ram ha un tempo di accesso di 1 microsecondo.

a) Se il sistema adotta la paginazione della memoria (ma non implementa la memoria virtuale), con le tabelle delle pagine mantenute in RAM, ed un TLB con miss ratio medio del 10% e tempo di accesso di 0,1 microsecondi, qual è l,'Effective Access Time (EAT) del sistema (esplicitate i vostri calcoli)?

0,90 * (1 + 0,1) + 0,1 *(2 + 0,1) = 0,90 + 0,09 + 0,2 + 0,01 = 1,2 microsecondi

b) Si supponga ora che il sistema implementi anche la memoria virtuale (e che, quando la ricerca nel TLB ha successo, la pagina riferita si trovi effettivamente in RAM – Il TLB ha ancora un miss ratio del 10%). Si sa che il 4% degli indirizzi logici generati provoca un page fault, e nel 50% dei casi di page fault la pagina vittima ha il dirty bit a 0. Il tempo medio necessario per trasferire una pagina dalla RAM all'area di swap (o viceversa) è di 1 millisecondo. Assumendo ora, per semplicità, un TLB con tempo di accesso pari 0, qual è l'EAT medio del sistema? (esplicitate i vostri calcoli, e specificate le eventuali assunzioni che fate)

Eat = (in microsecondi)
0.90 * 1 + (pagina nel TLB) (0.9)
0,06 * 2 + (pagina in RAM) (0,12)
0,02 * (1 + 1000 + 2) (page fault, dirty bit a 0) (20,06)
0,02 * (1 + 1000 * 2 + 2) (page fault, dirty bit a 1) (40,06)
= 61,12 microsecondi
(dove 1 è il microsecondo speso nell'accesso alla tabella delle pagine per accorgersi che la pagina indirizzata non è presente, e 2 sono i microsecondi necessari ad accedere alla cella indirizzata una volta che il page fault è stato servito. Questi valori possono anche essere ignorati, in quanto trascurabili rispetto al tempo necessario per accedere all'area di swap)

c) In un sistema operativo che implementa la memoria virtuale con paginazione su richiesta (demand paging) e con allocazione globale delle pagine, si osserva che la CPU è utilizzata in

media per il 20% del tempo, e il disco di swap è attivo in media per il 97% del tempo. Come è possibile spiegare questa situazione (motivate in maniera sintetica la vostra risposta)?

Il sistema è in trashing: i processi passano la maggior parte del tempo a rubarsi l'un l'altro le pagine, per cui ciascun processo passa la maggior parte del tempo attendendo l'arrivo della pagina mancante, che va prelevata dall'area di swap. La CPU rimane spesso inutilizzata mentre il disco che ospita l'area di swap è continuamente in uso.

d) Rispetto alla situazione della domanda c), elencate almeno due soluzioni che potrebbero migliorare l'utilizzo della CPU.

Diminuire il grado di multiprogrammazione, aumentare la dimensione della RAM, installare un HD per l'area di swap più veloce.

e) Un page fault medio del 4% degli indirizzi generati (come nel sistema considerato nella prima parte della domanda) è da ritenersi un valore, basso, accettabile, o eccessivo (motivate la vostra risposta)?

E' chiaramente un valore eccessivo, in quanto il tempo di accesso medio EFFETTIVO alla ram (EAT) peggiora di circa 60 volte rispetto al tempo di accesso di base di 1 microsecondo.

ESERCIZIO 3 (9 punti)

a) Un sistema operativo è in grado di decidere, scegliendo tra le tre modalità di base di allocazione dello spazio su disco, quella più adeguata per memorizzare un file in base alle seguenti informazioni, note al S.O. stesso: **(1)** numero di blocchi occupati dal file, **(2)** tipo di accesso al file (*sequenziale* o *diretto*, che viene dichiarato dall'utente all'atto della creazione del file stesso*)*.
Per ciascuno dei file riportati qui di seguito, indicate quale modalità di allocazione sceglierà il S.O.:

FileA: 1 blocco, sequenziale: contigua (o concatenata, l'indicizzata spreca spazio inutilmente)
FileB: 100 blocchi, diretto: indicizzata
FileC: 1 blocco, diretto: contigua (o concatenata, l'indicizzata spreca spazio inutilmente)
FileD: 100 blocchi, sequenziale: concatenata (e' ragionevole anche l'indicizzata, sebbene produca un maggiore spreco di spazio, perché è più affidabile della concatenata)

b) Nel sistema descritto nel punto a), i blocchi su disco occupano 512 byte, e un puntatore a blocco è scritto su 4 byte. Di un file si sa che deve essere acceduto in modo diretto. Quanti accessi al disco sono necessari per leggere direttamente il contenuto del blocco numero 200 del file, assumendo che gli attributi del file in questione siano già in memoria primaria? (motivate la vostra risposta).

3. Il S.O. userà l'allocazione indicizzata. In un blocco indice possiamo scrivere 128 puntatori, per cui un solo blocco indice non è sufficiente ad indirizzare il blocco 200. Sia usando l'allocazione indicizzata gerarchica che l'allocazione indicizzata concatenata, un secondo blocco indice è sufficiente per indirizzare il blocco 200. Sono quindi necessari due accessi al disco per leggere i due blocchi indice + un accesso per leggere il blocco 200.

c) Consideriamo una generica directory all'interno del file system del sistema. Quanti pathname *assoluti* sono associati a quella cartella? (motivate la vostra risposta).

La cartella possidede un pathname assoluto: il percorso dalla radice (root) del file system alla cartella stessa.

d) Nel sistema del punto a) si viene a sapere che tutti i file devono poter essere acceduti in modo diretto. Quale dei tre metodi di allocazione usati dal sistema verrebbe certamente abbandonato, e con quale "variante" verrebbe sostituito? Descrivete brevemente questa variante con l'esempio di un hard disk da 12 blocchi in cui è contenuto un unico file allocato, nell'ordine, nei blocchi 8, 5, 10, 1. (usate "-2" per indicare un blocco libero)

L'allocazione concatenata verrebbe sostituita dalla variante della FAT (la contigua può ancora essere usata per file che occupano uno o due blocchi).

-2	-1	-2	-2	-2	10	1	2	5	-2	1	-2
0	1	2	3	4	5	6	7	8	9	10	11

e) Cosa succede quando si esegue la *open* di un file, in un generico sistema operativo?

Si comunica al sistema che si vuole usare quel file: il SO recupera gli attributi del file e li copia in RAM, nella open file table. Successivi accessi agli attributi del file da parte del programma che ha chiamato la open useranno la copia degli attributi in RAM, anziché la copia su disco (ad accesso molto più lento). Di solito, anche una parte del file stesso viene copiata in RAM, in modo da velocizzare l'accesso ai dati del file. La open restituisce un *file pointer* che verrà usato dal programma per accedere ai vari dati/attributi del file disponibili in RAM.