Sistemi Operativi
1 settembre 2011
Compito

Si risponda ai seguenti quesiti, giustificando le risposte.

1. (a) Si discutano i criteri di valutazione degli algoritmi di scheduling della CPU.
   (b) L'algoritmo di scheduling RR è fair rispetto ai processi, ma non rispetto agli utenti. Come potrebbe essere modificato in modo da renderlo fair rispetto agli utenti?

   Risposta:

   (a) (3 punti) I criteri di valutazione degli algoritmi di scheduling della CPU sono i seguenti:
   - *utilizzo della CPU:* mantenere la CPU più carica possibile;
   - *throughput:* numero di processi completati nell'unità di tempo;
   - *tempo di turnaround:* tempo totale impiegato per l'esecuzione di un processo;
   - *tempo di attesa:* quanto tempo un processo ha atteso in coda ready;
   - *tempo di risposta:* quanto tempo si impiega da quando una richiesta viene inviata a quando si ottiene la prima risposta (non l'output);
   - *varianza del tempo di risposta:* quanto il tempo di risposta è variabile.

   (b) (2 punti) Per rendere l'algoritmo di scheduling RR fair rispetto agli utenti, sufficiente mantenere una coda di processi diversa per ogni utente. In tal modo il quanto di tempo viene assegnato all'utente e non al singolo processo. Quindi tutti i processi di un utente avrebbero a disposizione un solo quanto di tempo per ogni round dell'algoritmo.

2. Si consideri un sistema con scheduling SJF con prelazione (cioè SRTF), ove $\alpha = 0,3$ e $\tau_0 = 30$ msec. All'istante 0 il processore si libera e tre processi, P_1, P_2, P_3, sono in coda ready. Finora i processi P_1, P_2 sono andati in esecuzione due volte con CPU burst $20, 40$ msec per P_1 e $35, 20$ msec per P_2; mentre P_3 è andato in esecuzione una volta con CPU burst di 50 msec.
   Si determini:
   (a) Quale processo viene selezionato dallo scheduler all'istante 0?
   (b) All'istante 10 msec entra nella coda ready un nuovo processo P_4 con CPU burst previsto di 20 msec. Il processo selezionato precedentemente è ancora in esecuzione. Che cosa succede?
   (c) Che cosa succede quando il processo in esecuzione termina il suo burst?

   Risposta:

   (a) (3 punti) Per P_1 abbiamo $\tau_1 = 0,3 \cdot 20 + 0,7 \cdot 30 = 6 + 21 = 27$, $\tau_2 = 0,3 \cdot 40 + 0,7 \cdot 27 = 12 + 18,9 = 30,9$, per P_2 abbiamo $\tau_1 = 0,3 \cdot 35 + 0,7 \cdot 30 = 10,5 + 21 = 31,5$, $\tau_2 = 0,3 \cdot 20 + 0,7 \cdot 31,5 = 6 + 22,05 = 28,05$ ed infine per P_3 abbiamo $\tau_1 = 0,3 \cdot 50 + 0,7 \cdot 30 = 15 + 21 = 36$. Quindi all'istante 0 SRTF sceglie P_2.

   (b) (3 punti) All'istante 10 msec quando entra nella coda ready un nuovo processo P_4 con CPU burst previsto di 20 msec, continua P_2 perché gli mancano meno di 20 ms ($28,05 - 10 = 18,05$ ms) che è il burst previsto per P_4.

   (c) (3 punti) Quando P_2 termina, va in esecuzione P_4, dato che ha il burst previsto più piccolo.

3. Un computer ha 4 frame, i cui istanti di caricamento, ultimo riferimento e i reference bit sono riportati nella seguente tabella:

Frame	Caric.	Rifer.	R
0	140	269	1
1	251	260	1
3	172	287	0
2	219	304	1

   Dire quale pagina verrebbe liberata dall'algoritmo FIFO, da LRU e da CLOCK (in questo caso si supponga che l'ultimo frame controllato sia 2).

   Risposta: (3 punti) L'algoritmo FIFO, in base ai dati della tabella, avrà la coda 0, 3, 2, 1 e quindi sceglierà la pagina nel frame 0. LRU invece avrà la pila 2, 3, 0, 1, dove la pagina del frame 1 sarà quella riferita meno recentemente e quindi quella liberata dall'algoritmo. Infine CLOCK avrà la stessa coda di FIFO, ovvero, 0, 3, 2, 1 ma la gestirà in modo circolare, dando una seconda chance alle

pagine con reference bit impostato a 1. Quindi (siccome viene detto che l'ultimo frame controllato è stato il 2) viene controllato dapprima il frame 1: siccome il reference bit è 1, quest'ultimo viene impostato a 0 e la pagina del frame risparmiata. Il controllo prosegue quindi con il frame 0 che, avendo il reference bit impostato a 1, viene saltato (impostando il reference bit a 0). Si giunge così al frame 3 che, avendo il reference bit impostato a 0, viene scelto dall'algoritmo e la pagina che contiene viene scelta per la sostituzione.

4. Un sistema con risorse prelazionabili usa la seguente politica di allocazione delle risorse. Quando una risorsa richiesta da un processo P_i non è disponibile:

   1. la risorsa viene tolta ad uno dei processi che la detiene, P_j, se P_j è più recente di P_i, e viene allocata a P_i; viene restituita a P_j quando P_i la rilascia (un processo è *più recente* se ha iniziato dopo la computazione);
   2. se non c'è un processo più recente di P_i, allora P_i si blocca in attesa della risorsa.

   Una risorsa che viene rilasciata è sempre allocata al processo più vecchio tra quelli che l'hanno richiesta.

   Si dimostri che il deadlock non si può verificare in questo sistema.

   Risposta: (3 punti) In questo sistema non si può verificare il deadlock in quanto viene a mancare una delle quattro condizioni di Coffman, ovvero, la mancanza di prerilascio. Infatti le risorse possono essere tolte ai processi nel caso in cui ne faccia richiesta un processo più vecchio.

5. In merito alla protezione del filesystem, si spieghi brevemente:

   1. cos'è una *matrice di accesso*;
   2. cos'è una *access control list*;
   3. come viene implementata la protezione in UNIX.

   Risposta:

   1. (2 punti) Una *matrice di accesso* tiene traccia in ogni riga del dominio di protezione (e.g., utenti, processi) ed in ogni colonna dell'oggetto del filesystem da proteggere; quindi ad ogni coppia (processo,oggetto), associa le operazioni permesse.
   2. (2 punti) Dato che le matrici di accesso sono molto sparse, possono essere implementate come access control list: ad ogni oggetto, tali strutture associano chi (dominio) può fare cosa (operazione).
   3. (2 punti) La protezione in UNIX è implementata come una versione semplificata di ACL. Vi sono tre modi di accesso (operazioni consentite): read, write, execute. I domini consistono in tre classi di utenti per ogni file: proprietario, gruppo e resto degli utenti (ovvero, chi non rientra nei primi due domini). Per ogni classe di utenti una tripletta di bit specifica le operazioni consentite (0 indica operazione negata, 1 indica operazione consentita). Il primo bit corrisponde all'operazione read, il secondo alla write ed il terzo all'execute. Ogni utente (e quindi ogni processo dell'utente) possiede UID e GID, con i quali si verifica quale sia il dominio di protezione corrispondente e quindi la liceità dell'accesso.

6. Si consideri un file system UNIX-like (ad esempio, UFS o EXT2) con blocchi da 4KB, su un disco con $t_{seek} = 10ms$, a 7200 RPM (rotazioni per minuto). Si consideri inoltre il caso in cui in tale file system sia già stato aperto un file (quindi il relativo inode sia già stato caricato in memoria) i cui blocchi (sia indice che dati) risiedano sulla *stessa traccia*. Si supponga infine che i puntatori *diretti* nell'inode siano 12.

   1. Quanto si impiega *mediamente*[1] per *accedere* direttamente alla posizione 10000 del file?
   2. e alla posizione 100000?

[1] Sugg.: si consideri quindi una latenza media pari al tempo necessario a compiere mezza rotazione del disco.

Risposta:

1. (2 punti) La posizione 10000 cade nel terzo blocco, che è uno dei blocchi diretti. Per cui basta 1 accesso al disco (l'inode è già stato caricato in memoria al momento dell'apertura), che costa $t_{seek} + t_{latenza} = 10 + 60/(2 * 7, 2) = 14, 17msec.$
2. (2 punti) La posizione 100000 cade in uno dei primi indiretti, per cui è necessario accedere 2 volte al disco (una volta anche per il blocco indiretto), dove però il tempo di seek si conta una volta sola perché i blocchi sono sulla stessa traccia. In totale $10 + 4, 17 + 4, 17 = 18, 34msec.$