Università Ca' Foscari Venezia – Corso di Laurea in Informatica
Sistemi Operativi A

Problemi sulla gestione della memoria

Prof. Augusto Celentano, anno accademico 2007–2008

Problema 1

Un sistema a memoria paginata ha un tempo di accesso alla memoria primaria di 100 nanosecondi. Per gestire un page fault occorrono mediamente 10 mS se è disponibile una pagina vuota in memoria centrale, o se la pagina da sostituire non è modificata, e 24 mS se si deve sostituire una pagina modificata. Se nel 50% dei casi la pagina da sostituire è modificata, qual'è la probabilità accettabile di page fault per un tempo medio di accesso di 200 nanosecondi? Qual è il tempo medio di accesso per una probabilità di page fault di 10^{-4} ?

Soluzione

La formula generale è la seguente, dove *tm* è il tempo di accesso medio, *p* è la probabilità di page fault, *tc* è il tempo di accesso a memoria centrale, e *tf* è il tempo di servizio di un page fault (in cui si trascura il tempo di accesso a memoria centrale perché trascurabile rispetto all'accesso a disco):

$$tm = (1 - p) \times tc + p \times tf$$

Nel nostro caso, essendo la probabilità di sostituire una pagina modificata pari al 50% dei page fault, il tempo medio di servizio di un page fault è 17 mS (media tra 10 mS e 24 mS). Essendo *tm* = 200 nS e *tc* = 100 nS, si ha:

$$200 = (1 - p) \times 100 + p \times 17 \times 10^{6}$$

arrotondando, p = 1 / 170.000, cioè un page fault ogni 170.000 accessi a memoria.

Con la stessa formula si calcola il tempo medio di accesso per una probabilità di page fault di 10^{-4}:

$$tm = (1 - 10^{-4}) \times 100 + 10^{-4} \times 17 \times 10^{6} \sim= 100 + 1.700 = 1{,}8\ \mu S$$

Problema 2

La memoria virtuale di un calcolatore è divisa in pagine di 512 byte. Dati i seguenti frammenti di programma in linguaggio C:

```
int x[32][128];
int r, c;
...
for (r = 0; r < 32; r++)
    for (c = 0; c < 128; c++)
        x[r][c] = 0;
```

(a)

```
int x[32][128];
int r, c;
...
for (c = 0; c < 128; c++)
    for (r = 0; r < 32; r++)
        x[r][c] = 0;
```

(b)

dire quanti page fault vengono generati nell'ipotesi che

- per i dati vengano usati 8 page frame, inizialmente non allocati
- i dati di tipo `int` occupino ciascuno 4 byte
- la matrice sia memorizzata per righe in modo contiguo e compatto
- gli unici dati allocati in memoria paginata siano gli elementi della matrice
- si adotti una politica di sostituzione locale delle pagine LRU

Soluzione

Se ogni dato intero occupa 4 byte, ogni riga della matrice occupa 128 x 4 = 512 byte, cioè una pagina.

Nel caso (a) la matrice viene azzerata per righe, quindi una volta caricata in memoria la pagina corrispondente alla prima riga questa può essere azzerata senza ulteriori page fault. Quindi si carica in memoria la pagina che contiene la seconda riga, e così via fino all'ottava riga. A questo punto tutte le pagine fisiche a disposizione sono state occupate, e per azzerare la riga di indice 8 (cioè la nona) è necessario riutilizzare una pagina già utilizzata. In base alla politica LRU si riutilizza la prima pagina fisica allocata, che contiene la riga di indice 0, che è quella utilizzata meno recentemente. Si prosegue quindi riutilizzando la seconda pagina, poi la terza, fino all'ottava in cui viene allocata la riga di indice 15. Proseguendo nello stesso modo, si genera un page fault per ogni riga e quindi in totale si hanno 32 page fault.

Nel caso (b) la matrice viene azzerata per colonne, ma l'allocazione è sempre per righe. Viene quindi caricata la prima riga nella prima pagina, e viene azzerato l'elemento di indice [0][0]. Per azzerare l'elemento di indice [0][1] è necessario caricare la seconda riga, che occupa la seconda pagina fisica, e così via fino all'elemento di indice [0][7] che occupa l'ottava pagina. Per l'elemento di indice [0][8]è necessario liberare una pagina fisica. Come prima, per la politica LRU viene liberata la prima pagina allocata. Proseguendo in questo modo, risulta che per ogni elemento della matrice da azzerare deve essere allocata una nuova pagina, per un totale di 32 x 128 = 4096 page fault.

Problema 3

Un programma in formato eseguibile è diviso in tre parti: il codice, i dati e lo stack. Si supponga di avere un programma caratterizzato da un codice di 234 Kbyte, un'area dati di 313 Kbyte e uno stack di 64 Kbyte. Si ipotizzi che la macchina fisica possa avere 128 Megabyte di memoria massima teorica, che l'indirizzamento logico sia su 22 bit, e che il programma sia tutto in memoria durante l'esecuzione. La memoria è organizzata a pagine, con pagine di 8 Kbyte.

a. Quanto è lunga la tabella delle pagine per il programma?
b. Quanti bit ha ogni elemento della tabella?
c. Qual è il numero di pagine logiche del più lungo programma eseguibile?

Soluzione

Il programma occupa:

- per il codice $\lceil 234 / 8 \rceil = 30$ pagine,
- per i dati $\lceil 313 / 8 \rceil = 40$ pagine,
- per lo stack $\lceil 64 / 8 \rceil = 8$ pagine,

quindi complessivamente 78 pagine.

La memoria fisica è composta da 16.384 pagine (128Mbyte / 8Kbyte), quindi ogni elemento della tabella delle pagine ha 14 bit.

Utilizzando un metodo di calcolo alternativo, 128 Mbyte richiedono 27 bit, ogni pagina richiede 13 bit, quindi occorrono 14 bit per identificare una pagina fisica.

Il più lungo programma eseguibile (non in overlay) occupa tutto l'indirizzamento logico e occupa quindi 512 pagine (4Mbyte / 8Kbyte).

Usando un metodo di calcolo alternativo, l'indirizzamento logico su 22 bit riserva 9 bit al numero di pagina, con cui si indirizzano 512 pagine.

Problema 4

La tabella delle pagine di un processo in esecuzione è la seguente. Tutte le numerazioni sono decimali, il processo utilizza 3 pagine fisiche di 16 Kbyte l'una, i tempi sono espressi in unità convenzionali ed esprimono il tempo trascorso dal caricamento o dall'ultimo accesso alla pagina fisica.

# pag. virtuale	*valida*	*usata*	*modificata*	*# pag. fisica*	*tempo dal caricamento*	*tempo dall' ultimo uso*
0	1	1	0	12	127	123
1	1	1	0	5	130	130
2	1	1	1	3	143	112
3	0	0	0	-	-	-

Per ognuno dei seguenti indirizzi logici: 17.830, 36.215, 62.512, dire
- se un riferimento ad esso genera o no un page fault;
- se sì, qual è la pagina fisica sostituita da un algoritmo:
  - FIFO
  - LRU
  - Clock, supponendo che le pagine vengano esaminate nell'ordine delle righe della tabella a partire dalla prima riga

  e se la pagina vittima deve essere scritta su disco
- qual è l'indirizzo fisico corrispondente (eventualmente dopo la risoluzione del page fault)

Soluzione

Gli indirizzi indicati si trovano nelle seguenti pagine virtuali:

17.830 pagina virtuale 1, offset 1.446 (17.830 = 1 x 16.384 + 1.446)
36.215 pagina virtuale 2, offset 3.447 (36.215 = 2 x 16.384 + 3.447)
62.512 pagina virtuale 3, offset 13.360 (62.512 = 3 x 16.384 + 13.360)

Le pagine virtuali 1 e 2 hanno il bit di validità impostato a 1, quindi i primi due indirizzi sono presenti in memoria centrale. Le corrispondenze con gli indirizzi fisici sono le seguenti.

- indirizzo virtuale 17.830, pagina virtuale 1, pagina fisica 5: l'indirizzo fisico corrispondente è 5 x 16.384 + 1.446 = 83.366.
- indirizzo virtuale 36.215, pagina virtuale 2, pagina fisica 3: l'indirizzo fisico corrispondente è 3 x 16.384 + 3.447 = 52.599.

L'indirizzo virtuale 62.512 si trova nella pagina virtuale 3 che non è presente in memoria, come indicato dal bit di validità impostato a 0. Un riferimento a questo indirizzo genera quindi un page fault.

- Adottando una politica di sostituzione FIFO si utilizza la pagina fisica 3, che è stata caricata da più tempo. Essendo modificata deve essere scritta su disco. L'indirizzo fisico dopo la risoluzione del page fault è 3 x 16.384 + 13.360 = 62.512 (coincide con l'indirizzo logico perché la pagina logica coincide con quella fisica).
- Adottando una politica di sostituzione LRU si riutilizza la pagina fisica 5, che non viene utilizzata da più tempo. Non essendo modificata non deve essere scritta su disco. L'indirizzo fisico che ne risulta è 5 x 16.384 + 13.360 = 95.280.
- Con una politica di sostituzione Clock viene riutilizzata la pagina fisica 12. Infatti poiché tutte le pagine sono state utilizzate recentemente i loro bit di uso vengono rimessi a zero, quindi al secondo passaggio viene selezionata la pagina che occupa la prima posizione della tabella. La pagina non è modificata e quindi non deve essere scritta su disco. L'indirizzo fisico risultante è 12 x 16.384 + 13.360 = 209.968.