Esame 2021/06/25

Blascovich Alessio

Domanda 1

MAX PUNTI: 5
Domanda:
Spiegare nel dettaglio le differenze tra paginazione e frammentazione della memoria.
Soluzione:
Partono entrambe dallo stesso principio, ovvero quello che lo spazio dato ad un processo non deve per forza essere contiguo:

- *Paginazione*: divide la memoria fisica in frame e quella logica in pagine, in pratica l'indirizzo logico viene visto come diviso in due parti: il numero della pagina e l'offset dall'inizio della pagina.
  Ogni frame ha dimensione fissa, ma grazie alla struttura dell'indirizzo logico posso andare in cerca della pagina prima nel TLB poi nella tabella delle pagine, una volta trovato l'intervallo di memoria fisica rappresentato da quella pagina posso sommare l'offset e ottenere l'indirizzo fisico.
  In generale però usando la paginazione anche se si possono dividere le aree di memoria questo non viene quasi mai fatto, infatti solitamente processi diversi hanno indirizzi logici e fisici diversi.

- *Segmentazione*: la memoria viene divisa come l'utente la vedrebbe.
  Un programma è diviso in segmenti, ognuno dei quali contiene informazioni come: il main, funzioni, variabili e stack.
  Così facendo l'indirizzo logico è formato dal numero del segmento e da un offset, che può sfruttare il TLB per velocizzare la ricerca.
  La traduzione avviene in modo analogo a quella della paginazione, solo che con la segmentazione è più facile condividere codice come quello delle librerie.

Domanda 2

MAX PUNTI: 6
Domanda:
Un calcolatore riceve una sequenza di richieste a memoria nel seguente ordine 1,4,3,3,9,7,5,2,3,2,3,2,2,5,7,6,9,2,9,2,3,1,2. Il calcolatore preso in esame ha 4 frame per la memoria(inizialmente vuoti), calcolare il numero di page fault con gli algoritmi:

1. *LRU*
2. *Seconda opportunità*

Soluzione:
Si consideri l'acronimo pf usato di seguito come l'abbreviazione di page fault.

1. *LRU*
   L'*LRU* è un algoritmo basato su un contatore, ovvero ad ogni frame viene assegnato il contatorePrecedente+1.
   Quando si deve togliere un frammento per fare spazio ad un altro si scarta quello con il contatore più basso, ovvero quello che non viene usato da più tempo.
   Nelle seguenti tabelle la prima riga indica il contatore, mentre la seconda il numero del segmento.

1	null	null	null
1	null	null	null

pf=1 ⟹

1	2	null	null
1	4	null	null

pf=2 ⟹

1	2	3	null
1	4	3	null

pf=3 ⟹

1	2	4	null
1	4	3	null

pf=3 ⟹

1	2	4	5
1	4	3	9

pf=4 ⟹

6	2	4	5
7	4	3	9

pf=5 ⟹

6	7	4	5
7	5	3	9

pf=6 ⟹

6	7	8	5
7	5	2	9

pf=7 ⟹

6	7	8	9
7	5	2	3

pf=8 ⟹

6	7	10	9
7	5	2	3

pf=8 ⟹

6	7	10	11
7	5	2	3

pf=8 ⟹

6	7	12	11
7	5	2	3

pf=8 ⟹

6	7	13	11
7	5	2	3

pf=8 ⟹

6	14	13	11
7	5	2	3

pf=8 ⟹

15	14	13	1
7	5	2	3

$\xrightarrow{\text{pf=8}}$	15	14	13	16
7	5	2	6

$\xrightarrow{\text{pf=9}}$	15	14	17	16
7	5	9	6

$\xrightarrow{\text{pf=10}}$	15	18	17	16
7	4	9	6

$\xrightarrow{\text{pf=11}}$	15	18	17	19
7	4	9	2

$\xrightarrow{\text{pf=12}}$	15	18	20	19
7	4	9	2

$\xrightarrow{\text{pf=12}}$	15	18	20	21
7	4	9	2

$\xrightarrow{\text{pf=12}}$	22	18	20	21
3	4	9	2

$\xrightarrow{\text{pf=13}}$	22	23	20	21
3	1	9	2

$\xrightarrow{\text{pf=14}}$	22	23	20	24
3	1	9	2

$\xrightarrow{\text{pf=14}}$ Termino con un totale di 14 page fault.

2. *Seconda opportunità*:
   Questo algoritmo si basa su una struttura FIFO, ad ogni frame viene assegnato un bit il quale inizialmente è 0, mentre se la cella viene richiamata diventa 1.
   Quando la memoria è satura devo prendere il frame più vecchio(FIFO) e controllare il bit, se questo è 0 rimpiazzo il frame altrimenti diminuisco il bit da 1 a 0 e vado al frame prossimo.
   Nel malaugurato caso tutti i bit fossero 1 ciclo fino a farli diventare tutti 0 e poi sostituisco il frame più vecchio.
   Se la spiegazione non dovessere essere stata chiara qui un video dove mostrano un esercizio spiegato passo passo.
   La prima riga delle tabelle indica lo stato del bit, mentre la seconda il numero del frame, per facilitare la lettura le celle saranno fatte scorrere come una coda che ha il primo elemento nella prima cella a partire da sinistra, consiglio di farlo in compito altrimenti con questo algoritmo non si tendono a fare molti errori.

0	null	null	null
1	null	null	null

$\xrightarrow{\text{pf=1}}$ | 0 | 0 | null | null |
|---|---|---|---|
| 1 | 4 | null | null |

$\xrightarrow{\text{pf=2}}$ | 0 | 0 | 0 | null |
|---|---|---|---|
| 1 | 4 | 3 | null |

$\xrightarrow{\text{pf=3}}$ | 0 | 0 | 1 | null |
|---|---|---|---|
| 1 | 4 | 3 | null |

$\xrightarrow{\text{pf=3}}$ | 0 | 0 | 1 | 0 |
|---|---|---|---|
| 1 | 4 | 3 | 9 |

$\xrightarrow{\text{pf=4}}$ | 0 | 1 | 0 | 0 |
|---|---|---|---|
| 4 | 3 | 9 | 7 |

$\xrightarrow{\text{pf=5}}$ | 1 | 0 | 0 | 0 |
|---|---|---|---|
| 3 | 9 | 7 | 5 |

$\xrightarrow{\text{pf=6}}$ | 0 | 0 | 0 | 0 |
|---|---|---|---|
| 3 | 7 | 5 | 2 |

$\xrightarrow{\text{pf=7}}$ | 1 | 0 | 0 | 0 |
|---|---|---|---|
| 3 | 7 | 5 | 2 |

$\xrightarrow{\text{pf=7}}$ | 1 | 0 | 0 | 1 |
|---|---|---|---|
| 3 | 7 | 5 | 2 |

$\xrightarrow{\text{pf=7}}$ | 1 | 0 | 0 | 1 |
|---|---|---|---|
| 3 | 7 | 5 | 2 |

$\xrightarrow{\text{pf=7}}$ | 1 | 0 | 0 | 1 |
|---|---|---|---|
| 3 | 7 | 5 | 2 |

$\xrightarrow{\text{pf=7}}$ | 1 | 0 | 1 | 1 |
|---|---|---|---|
| 3 | 7 | 5 | 2 |

$\xrightarrow{\text{pf=7}}$ | 1 | 1 | 1 | 1 |
|---|---|---|---|
| 3 | 7 | 5 | 2 |

$\xrightarrow{\text{pf=7}}$ | 0 | 0 | 0 | 0 |
|---|---|---|---|
| 7 | 5 | 2 | 6 |

$\xrightarrow{\text{pf=8}}$ | 0 | 0 | 0 | 0 |
|---|---|---|---|
| 5 | 2 | 6 | 9 |

$\xrightarrow{\text{pf=9}}$ | 0 | 1 | 0 | 0 |
|---|---|---|---|
| 5 | 2 | 6 | 9 |

$\xrightarrow{\text{pf=9}}$ | 0 | 1 | 0 | 1 |
|---|---|---|---|
| 5 | 2 | 6 | 9 |

$\xrightarrow{\text{pf=9}}$ | 0 | 1 | 0 | 1 |
|---|---|---|---|
| 5 | 2 | 6 | 9 |

$\xrightarrow{\text{pf=9}}$ | 1 | 0 | 1 | 0 |
|---|---|---|---|
| 2 | 6 | 9 | 3 |

$\xrightarrow{\text{pf=10}}$ | 0 | 1 | 0 | 0 |
|---|---|---|---|
| 2 | 9 | 3 | 1 |

$\xrightarrow{\text{pf=11}}$ | 1 | 1 | 0 | 0 |
|---|---|---|---|
| 2 | 9 | 3 | 1 |

$\xrightarrow{\text{pf=11}}$ Termino con un totale di 11 page fault.

Domanda 3

MAX PUNTI: 6
Domanda:

Data la tabella sottostante si applichino gli algoritmi di scheduling *HRRN* e *RR* con quanto uguale a 2 e si calcolino $T_{risposta}$, T_{attesa} e $T_{turnaround}$.

processo	T_{burst}	T_{arrivo}
1	3	0
2	1	1
3	2	3
4	4	4
5	8	1

Soluzione:

Prima diamo un po' di definizioni:

- T_{attesa} non è altro che il tempo che un processo rimane nella coda dei processi pronti prima di essere eseguito una prima volta.
- $T_{risposta}$ il tempo totale che il processo è rimasto nella coda dei processi pronti.
- $T_{turnaround}$ il tempo passato tra la terminazione del processo e la sua entrata in coda.

Adesso inizio ad applicare gli algoritmi:

- *HRRN*:E' un algoritmo a priorità non-preemptive, la priorità viene calcolata come $P = 1 + \frac{T_{attesa}}{T_{burst}}$.
  Questo valore P va ricalcolato ogni qualvolta un processo termina la propria esecuzione oppure arriva un nuovo processo nella coda.

Processo	t=0	t=3	t=4	t=6	t=14	t=18
1	$1+\frac{0}{3}=1$					
2		$1+\frac{2}{1}=3$				
3		$1+\frac{0}{2}=1$	$1+\frac{1}{2}=\frac{3}{2}$			
4			$1+\frac{0}{4}=1$	$1+\frac{2}{4}=\frac{3}{2}$	$1+\frac{10}{4}=\frac{7}{2}$	fine
5		$1+\frac{2}{8}=\frac{5}{4}$	$1+\frac{3}{8}=\frac{11}{8}$	$1+\frac{5}{8}=\frac{13}{8}$		

Calcolo ora i vari tempi

Processo	$T_{risposta}$	T_{attesa}	$T_{turnaround}$
1	0	0	3
2	2	2	3
3	1	1	3
4	10	10	14
5	5	5	13

- *RR(Round robin)*: algoritmo basato su un time-out solitamente chiamato quanto.
  Il quanto è l'unità di tempo secondo la quale un processo può utilizzare la CPU, se un processi finisce prima si carica subito un altro processo, mentre se deve ancora terminare lo si accoda come in una *FCFS*.
  Come nella *FCFS* la priorità è data in base all'arrivo.

Tabella della coda di processi
Quando una cella è rossa indica che in quel quanto il processo termina

Exec	1	2	5	1	3	4	5	4	5
Queue	2	5	1	3	4	5	4	5	
	5	1	3	4	5				
		4	5						

Tabella dell'esecuzione dei processi
Il colore rosso indica quando un processo detiene la CPU.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1																		
2																		
3																		
4																		
5																		

Calcolo ora i vari tempi

Processo	$T_{risposta}$	T_{attesa}	$T_{turnaround}$
1	0	3	6
2	1	1	2
3	3	3	5
4	4	6	10
5	2	9	17

Domanda 4

MAX PUNTI: 6
Domanda:

a Spiegare nel dettaglio come è implementata la tabella a pagine invertite.

b Si consideri un calcolatore che usa un sistema di paginazioen a 2 livelli le cui tabelle siano mantenute in memoria principale.
Il tempo di accesso alla memoria è circa $T = 50$ns, usando una TLB con tempo di accesso pari a $T_{TLB} = 2$ns, quale hit ratio(α) dovremmo avere per poter mantere il degrado delle prestazioni al di sotto del 10% rispetto a T?(scrivere la formula per esteso)

Soluzione:

a Viene creata una tabella unica per tuttto il sistema, avente un elemento per ogni frame fisico che esiste.
Ogni elemento della tabella contiene la coppia <PID,numero-pagina>.

- *PID*: è l'id del processo che detiene la pagina.
- *numero-pagina*: contiene l'indirizzo logico della pagina che va a puntare al frame cercato.

Gli indirzzi generati dalla CPU devono essere quindi una tripla <PID,numero-apgina,offset>, i primi due attributi servono èper la ricerca nella tabella(sequenziale quindi O(n)), trovata la entry nella tabella si usa l'indice i della entry e lo si somma all'offset.
La ricerca potrebbe diventare se si usasse una hashmap perchè la ricerca avrebbe costo O(1) ma si dovrebbe implementare un metodo per gestire le collisioni.

b Scrivo la formula per esteso e poi la risolvo in α:

$$(T + T_{TLB}) * \alpha + ((3 * T) + T_{TLB}) * (1 - \alpha) \leq (T * 10\%)$$
$$(52) * \alpha + (152) * (1 - \alpha) \leq 5$$
$$52\alpha + 152 - 152\alpha \leq 5$$
$$-100\alpha \leq -147$$
$$\alpha \geq \frac{147}{100}$$
$$\alpha \geq 14.7\%$$

Domanda 5

MAX PUNTI: 8
Domanda:

In una mensa universitaria, dopo aver mangiato, gli studenti ripongono i vassoi in `M` contenitori, ognuno con `K` ripiani, un cameriere è addetto allo svuotare i contenitori di vassoi per poi riportarli in sala vuoti.
Scrivere un algoritmo che regoli l'accesso alle risorse minimizzando i tempi di attesa.
Soluzione:

```
int M,K;
int[][] contenitore=new int[0]*([M][K]);
Semafori_bin[] rack=Semafori_bin[1]*M;
bool[] pieno=new bool[0]*M;

Studente(){
   while(true){
      bool finito=false;
      for(int i=0;i<M && !finito;++i){
         P(rack[i]);
         if(pieno[i]==false){
            for(int j=0;j<K && !finito;++j){
               if(contenitori[i][j]!=0){
                  //metti vassoio
                  finito=true;
                  if(j==k-1){
                     pieno[i]=true;
                  }
               }
            }
         }
         V(rack[i]);
      }
   }
}

Cameriere(){
   for(int i=0;i<M;++i){
      P(rack[i]);
```

```
        if(pieno[i]==true){
            //svuota rack
            pieno[i]=false;
            contenitore[i]=[0]*K;
        }
        V(rack[i]);
        if(i==M-1){
            i=-1;
        }
    }
}
```