Eserciziario

Leonardo Ranaldi

January 2019

1 Modello di multiprogrammazione

1.1 Esercizio

In un computer con 1 GB di memoria, il sistema operativo occupa 512 MB e i processi occupano mediamente 128 MB, se l'attesa media dell'I/O è del 80%, qual è l'utilizzo della CPU? Aggiungendo 512 MB di RAM quanto sarà il nuovo utilizzo della CPU?

Soluzione

- Il tempo di attesa del 80% è riferito all'attesa di completamento di operazioni di I/O.
- Se abbiamo n processi in memoria e il tempo di attesa medio è p, allora la probabilità che tutti i processi siano in attesa (e quindi la CPU sia intattiva) è p^n.
- L'utilizzo della CPU è, dunque, $1 - p^n$.
- Il numero n di processi è ottenibile dividendo la dimensione di RAM libera per lo spazio occupato naturalmente da un solo processo.
- p = 80%
- n = (1GB – 512 MB) / 128 MB = 512/128 = 4
- Utilizzo CPU = $1 - p^n = 1 - 0.8^4 = 0.59$
- p = 80%
- Nuova RAM: 512MB
- n = (1GB – 512 MB+512 MB) / 128 MB = 1024/128 = 8
- Utilizzo CPU = $1 - p^n$ = 1–0.8^8 = 0.8322

1.2 Esercizio

In un computer con 1 GB di memoria, il sistema operativo occupa 512 MB e i processi occupano mediamente 64 MB, se l'attesa media dell'I/O è del 60%, qual è l'utilizzo della CPU? Aggiungendo 256 MB di RAM, quale sarà il nuovo utilizzo della CPU?

Soluzione

- Il tempo di attesa del 60% è riferito all'attesa di completamento di operazioni di I/O.
- Se abbiamo n processi in memoria e il tempo di attesa medio è p, allora la probabilità che tutti i processi siano in attesa (e quindi la CPU sia intattiva) è p^n.
- L'utilizzo della CPU è, dunque, $1 - p^n$.
- Il numero n di processi è ottenibile dividendo la dimensione di RAM libera per lo spazio occupato naturalmente da un solo processo.
- p = 60%
- n = (1GB – 512 MB) / 64 MB = 512/64 = 8
- Utilizzo CPU = 1- $p^n = 1 - 0.6^8 = 0.983$
- p = 60%
- Nuova RAM: 256MB
- n = (1GB – 512 MB+256) / 64 MB = 768/64 = 12
- Utilizzo CPU = 1- $p^n = 1 - 0.6^{12} = 0.997$

1.3 Esercizio

In un computer con 1 GB di memoria, il sistema operativo occupa 512 MB e i processi occupano mediamente 64 MB, se l'attesa media dell'I/O è del 65%, qual è l'utilizzo della CPU? Aggiungendo 512 MB di RAM quanto sarà il nuovo utilizzo della CPU?

Soluzione

- Il tempo di attesa del 65% è riferito all'attesa di completamento di operazioni di I/O.

- Se abbiamo n processi in memoria e il tempo di attesa medio è p, allora la probabilità che tutti i processi siano in attesa (e quindi la CPU sia intattiva) è p^n.
- L'utilizzo della CPU è, dunque, $1 - p^n$.
- Il numero n di processi è ottenibile dividendo la dimensione di RAM libera per lo spazio occupato naturalmente da un solo processo.
- p = 65%
- n = (1GB – 512 MB) / 64 MB = 512/64 = 8
- Utilizzo CPU = $1 - p^n = 1 - 0.65^8 = 0.968$
- p = 65%
- Nuova RAM: 512MB
- n = (1GB – 512 MB+512 MB) / 64 MB = 1024/64 = 16
- Utilizzo CPU = 1- p^n = 1–0.65^{16} = 0.998

2 Soft real-time con eventi periodici

2.1 Esercizio

Supponendo di dover valutare la fattibilità di un sistema soft real-time con eventi periodici $P_0 = 500\mu s$, $P_1 = 300\mu s$, $P_2 = 15ms$, $P_3 = 800ms$.
Con rispettivi tempi di elaborazione $C_0 = 100\mu s$, $C_1 = 100\mu s$, $C_2 = 3ms$, $C_3 = 200ms$.
Il sistema è sostenibile? Se si aggiunge un nuovo evento periodico $P_4 = 100ms$, con $C_4 = 2ms$. Il sistema rimane sostenibile? Perché?

Soluzione

Deadline: tempo entro il quale le operazioni devono necessariamente essere completate.
- Sistemi non obbligatoriamente veloci ma piuttosto affidabili.
- Eventi periodici: eseguiti a ciclo continuo con cadenza ben definita.
- Eventi periodici hanno:

- Periodo: tempo entro il quale l'evento deve essere ripetuto (Pi)
- Tempo d'esecuzione (Ci)
- Sistema Real Time sostenibile (con m eventi periodici): vale

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

Procediamo con la sostituzione dei valori all'interno della formula:

$$\frac{C_0}{P_0} + \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} \leq 1$$

Sostituendo in modo opportuno si ottiene:

$$\frac{100}{500} + \frac{100}{300} + \frac{3}{15} + \frac{200}{800} = \frac{59}{60} \leq 1$$

(Riferimento testo: Moderni sistemi operativi, paragrafo 2.5.4. Schedulazione nei sistemi real time, pag 135 e successive)

Massimo tempo di esecuzione per il nuovo processo P_4.

$$\frac{C_0}{P_0} + \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} + \frac{C_4}{P_4} \leq 1$$

Dato P_4 e C_4 sostituire nuovamente i valori e verificare il risultato.

$$\frac{100}{500} + \frac{100}{300} + \frac{3}{15} + \frac{200}{800} + \frac{2}{4} = \frac{89}{60} \leq 1$$

Ma attenzione perchè:

$$\frac{89}{60} \not\leq 1$$

Si può concludere quindi che il sistema non rimane sostenibile.

2.2 Esercizio

Supponendo di dover valutare la fattibilità di un sistema soft real-time con eventi periodici $P_0 = 300\mu s$, $P_1 = 900\mu s$, $P_2 = 45ms$, $P_3 = 150ms$.
Con rispettivi tempi di elaborazione $C_0 = 25\mu s$, $C_1 = 300\mu s$, $C_2 = 9ms$, $C_3 = 50ms$.
Il sistema è sostenibile? Se si aggiunge un nuovo evento periodico $P_4 = 110ms$, quanto è il tempo massimo di elaborazione affinché il sistema rimanga sostenibile?

Soluzione

Deadline: tempo entro il quale le operazioni devono necessariamente essere completate.
- Sistemi non obbligatoriamente veloci ma piuttosto affidabili.
- Eventi periodici: eseguiti a ciclo continuo con cadenza ben definita.
- Eventi periodici hanno:

- Periodo: tempo entro il quale l'evento deve essere ripetuto (Pi)

- Tempo d'esecuzione (Ci)
- Sistema Real Time sostenibile (con m eventi periodici): vale

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

Procediamo con la sostituzione dei valori all'interno della formula:

$$\frac{C_0}{P_0} + \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} \leq 1$$

Sostituendo in modo opportuno si ottiene:

$$\frac{25}{300} + \frac{300}{900} + \frac{9}{45} + \frac{50}{150} = \frac{57}{60} \leq 1$$

(Riferimento testo: Moderni sistemi operativi, paragrafo 2.5.4. Schedulazione nei sistemi real time, pag 135 e successive)

Massimo tempo di esecuzione per il nuovo processo P_4.

$$\frac{C_0}{P_0} + \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} + \frac{C_4}{P_4} \leq 1$$

Dato P_4 risolvere la disequazione precedente per trovare il valore di C_4.

$$\frac{57}{60} + \frac{C_4}{100} \leq 1$$

$$\frac{C_4}{100} \leq 1 - \frac{57}{60}$$

$$C_4 \leq \frac{3}{60} 110$$

$$C_4 \leq 5.5$$

Massimo tempo di esecuzione: 5.5ms

2.3 Esercizio3

Supponendo di dover valutare la fattibilità di un sistema soft real-time con eventi periodici $P_0 = 500\mu s$, $P_1 = 300\mu s$, $P_2 = 15ms$, $P_3 = 500ms$.
Con rispettivi tempi di elaborazione $C_0 = 50\mu s$, $C_1 = 50\mu s$, $C_2 = 3ms$, $C_3 = 100ms$.
Il sistema è sostenibile? Se si aggiunge un nuovo evento periodico $P_4 = 100ms$,

con $C_4 = 2ms$. Il sistema rimane sostenibile? Perché?

Soluzione

Deadline: tempo entro il quale le operazioni devono necessariamente essere completate.
- Sistemi non obbligatoriamente veloci ma piuttosto affidabili.
- Eventi periodici: eseguiti a ciclo continuo con cadenza ben definita.
- Eventi periodici hanno:

- Periodo: tempo entro il quale l'evento deve essere ripetuto (Pi)
- Tempo d'esecuzione (Ci)
- Sistema Real Time sostenibile (con m eventi periodici): vale

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

Procediamo con la sostituzione dei valori all'interno della formula:

$$\frac{C_0}{P_0} + \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} \leq 1$$

Sostituendo in modo opportuno si ottiene:

$$\frac{50}{500} + \frac{50}{300} + \frac{3}{15} + \frac{100}{500} = \frac{20}{30} \leq 1$$

(Riferimento testo: Moderni sistemi operativi, paragrafo 2.5.4. Schedulazione nei sistemi real time, pag 135 e successive)

Massimo tempo di esecuzione per il nuovo processo P_4.

$$\frac{C_0}{P_0} + \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} + \frac{C_4}{P_4} \leq 1$$

Dato P_4 e C_4 sostituire nuovamente i valori e verificare il risultato.

$$\frac{50}{500} + \frac{50}{300} + \frac{3}{15} + \frac{100}{500} + \frac{2}{4} = \frac{45}{30} \leq 1$$

Ma attenzione perchè:

$$\frac{45}{30} \not\leq 1$$

Si può concludere quindi che il sistema non rimane sostenibile.

3 Scheduling nei sistemi batch

3.1 Esercizio

Supponendo di utilizzare l'algoritmo shortest job first per lo scheduling in un sistema batch, se arrivano 4 job (in ordine A, B, C e D) con i seguenti tempi di esecuzione A=8, B=4, C=2 e D=4 quali sono i tempi di turnaround e quale il tempo medio?

Soluzione

- Shortest job firs: Viene eseguito il job minore per primo.
- Tempo turnaround: tempo che intercorre dal momento in cui un processo entra in coda per essere eseguito fino al termine della sua esecuzione.
- Tempo di attesa: tempo che un processo trascorre in coda prima di essere eseguito.
- **TA** (Time of arrival) tempo assoluto di ingresso in coda di un processo.
- **TE** (Time of execution) tempo necessario per l'esecuzione di un processo.

Tabella con TA e TE di ogni processo dell'esercizio.

Processo	TA	TE
A	0	8
B	0	4
C	0	2
D	0	4

- Strategia SJF: ordine C-B-D-A.
- TS = Tempo assoluto di inizio di esecuzione di ogni processo.
- TF = tempo assoluto di fine dell'esecuzione di un processo.

Processo	TA	TE	TS	TF
A	0	8	10	18
B	0	4	2	6
C	0	2	0	2
D	0	4	6	10

- TT = Tempo Turnaround = TF-TA
- TW = Tempo di attesa = TS-TA

Processo	TA	TE	TS	TF	TW	TT
A	0	8	10	18	10	18
B	0	4	2	6	2	6
C	0	2	0	2	0	2
D	0	4	6	10	6	10

- Tempo medio (entambi i casi): somma tempi e divisione per numero processi.

$$TW(medio) = \frac{10 + 2 + 0 + 6}{4} = 4,5$$

$$TT(medio) = \frac{18 + 6 + 2 + 10}{4} = 9$$

3.2 Esercizio

Supponendo di utilizzare l'algoritmo first-come first-served per lo scheduling in un sistema batch, se arrivano 4 job (in ordine A, poi B dopo 2ms, C dopo 4ms e D dopo 5ms) con i seguenti tempi di esecuzione A=8, B=4, C=2 e D=4 quali sono i tempi di turnaround? Qual è il tempo medio di turnaround e di attesa?

Soluzione

- First come – First served: Ordine in cui chi arriva prima viene eseguito per primo.
- Tempo turnaround: tempo che intercorre dal momento in cui un processo entra in coda per essere eseguito fino al termine della sua esecuzione.
- Tempo di attesa: tempo che un processo trascorre in coda prima di essere eseguito.
- **TA** (Time of arrival) tempo assoluto di ingresso in coda di un processo.
- **TE** (Time of execution) tempo necessario per l'esecuzione di un processo.

Tabella con TA e TE di ogni processo dell'esercizio.

Processo	TA	TE
A	0	8
B	2	4
C	4	2
D	5	4

- Strategia FCFS (FIFO): ordine A-B-C-D.
- TS = Tempo assoluto di inizio di esecuzione di ogni processo.

- TF = tempo assoluto di fine dell'esecuzione di un processo.

Processo	TA	TE	TS	TF
A	0	8	0	8
B	2	4	8	12
C	4	2	12	14
D	5	4	14	18

- TT = Tempo Turnaround = TF-TA
- TW = Tempo di attesa = TS-TA

Processo	TA	TE	TS	TF	TW	TT
A	0	8	0	8	0	8
B	2	4	8	12	6	10
C	4	2	12	14	8	10
D	5	4	14	18	9	13

- Tempo medio (entambi i casi): somma tempi e divisione per numero processi.

$$TW(medio) = \frac{0+6+8+9}{4} = 5.75$$

$$TT(medio) = \frac{8+10+10+13}{4} = 10.25$$

3.3 Esercizio

Supponendo di utilizzare l'algoritmo round–robin per lo scheduling in un sistema interattivo, se la coda è fatta dai processi A-B-C-D-E, nel caso in cui il quanto sia di 100 ms e il un tempo di cambio di contesto di 1 ms, quanto tempo sarà necessario prima che E venga eseguito? Qual è il rapporto tra cambio di contesto e tempo di esecuzione? Come cambiano I tempi analizzati nel caso in cui il quanto sia, invece, di 4 ms? Quale delle due soluzioni sembra più favorevole?

Soluzione

Round- robin:
Ad ogni processo viene assegnato un quanto di tempo in cui può andare in esecuzione, i processi vengono assegnati alla CPU tramite una coda circolare.
Se ha una richiesta bloccante o è terminato prima dello scadere del quanto, la CPU viene immediatamente assegnata al processo successivo.
Al termine di un quanto di tempo, viene cambiato l'assegnamento alla CPU, e deve intercorrere un certo tempo, il tempo di cambio di contesto prima che il nuovo processo possa iniziare la propria esecuzione.

- **E**: devono essere eseguiti per il loro quanto A, B, C e D e i ripettivi cambi di contesto. Quindi E deve aspettare, per la sua prima parte di esecuzione:

$$100 + 1 + 100 + 1 + 100 + 1 + 100 + 1 = 404ms$$

(quasi mezzo secondo)

- **Il rapporto è molto semplice:** 1ms/100ms = 0.01, (l'1%)

- **Secondo caso:**

$$4 + 1 + 4 + 1 + 4 + 1 + 4 + 1 = 20ms$$

- **Il rapporto è:** 1ms/4ms = 0.25, (il 25%)

- **Confronto:** pro prima soluzione:
  1. Tempo cambio di contensto molto più piccolo di tempo esecuzione;
  2. Processi con lunghe CPU burst favoriti;
- **Confronto:** contro prima soluzione:
  1. Tempo attesa processi potenzialmente lungo (400 ms è un tempo considerevole);
  2. Se frequenti I/O burst, simile a seconda soluzione;
- **Confronto:** pro seconda soluzione:
  1. Processi hanno tempo di attesa basso, importante specialmente se forniti di iterfaccia utente;
- **Confronto:** contro seconda soluzione:
  1. Cambio contesto è rilevante: confrontabile con tempo esecuzione;
  2. Sfavorisce processi con lunghe CPU burst;

4 Algoritmi di sostituzione delle pagine

4.1 Esercizio

Supponendo di utilizzare l'algoritmo di sostituzione delle pagine Not Recently Used (NRU), se la situazione è la seguente: Classe 1 = pag.2, Classe 0 = pag.3, Classe 3 = pag.4, Classe 2 = pag.5 Specificare quale pagina verrà posta su disco se accade un page fault spiegando le ragioni.

Soluzione
La maggior parte dei computer con memoria virtuale prevede 2 bit per la raccolta di informazioni sull'utilizzo delle pagine:

- il bit R: indica che la pagina è stata referenziata (letta o scritta)
- il bit M: indica che la pagina è stata modificata (scritta)

Algoritmo NRU (Not Recently Used)

1. Inizialmente i bit R e M vengono impostati a 0 dal SO
2. Periodicamente (ad esempio ad ogni interrupt del clock), il bit R viene azzerato per distinguere le pagine non referenziate recentemente dalle altre.

Quando avviene un page fault, il SO controlla tutte le pagine e le divide in quattro classi in base al valore corrente dei bit R e M.

Classe	R	M	Osservazioni
1	0	0	classe più conveniente da swappare
2	0	1	
3	1	0	
4	1	1	classe meno conveniente da swappare

L'algoritmo NRU rimuove una pagina qualsiasi (a caso) dalla classe di numero inferiore che sia non vuota.

Caratteristiche di NRU: è facile da implementare ed ha prestazioni che, anche se non ottimali, sono spesso adeguate.

Dopo una breve introduzione possiamo dire che in caso di page fault la pag.3 verrà scartat poichè è a livello più basso di tutti mentre la pagina 4 verrà posta sul disco poichè appartiene alla classe maggiore.

4.2 Esercizio

Nell'ambito del rimpiazzamento delle pagine, supponendo di usare l'algoritmo not recently used (NRU), e seguendo la tabella delle pagine in figura, nel caso in cui avvenga un page fault, quale sarà la pagina che verrà posta su disco? Perché? Se invece considerassimo la tabella come una lista ordinata (nell'ordine A,...,G), cosa succederebbe per l'algoritmo second chance? Quale pagina sarebbe rimpiazzata?

Pagina	R	M
A	1	0
B	1	1
C	1	0
D	0	1
E	0	0
F	1	1
G	1	0

Soluzione

- Bit R: posto a 1 ogni qual volta pagina sia riferita (cioè scritta o letta).
- Bit M: posto a 1 ogni qual volta pagina sia modificata.
- Reset a 0 da software per entrambi i bit.
- Priorità :

Classe	R	M	Osservazioni
1	0	0	classe più conveniente da swappare
2	0	1	
3	1	0	
4	1	1	classe meno conveniente da swappare

ALGORITMO SECOND CHANCE Pagine in lista per ordine di ingresso.
Bit M non considerato.

Algoritmo:

- Scorrere la lista
- Se R = 1, porre R=0, reinserire pagina in fondo a lista;
- Se R=0, effettuare swap della pagina corrispondente.
- Soluzione algoritmo LRU:
- Una sola pagina appartiene alla Classe 0, con massima priorità di swap.
- La pagina è E (M=0, R=0).
- Soluzione algoritmo second chance:
- A ha R=1: reset a 0 e spostamento in fondo alla lista,

- B ha R=1: reset a 0 e spostamento in fondo alla lista,
- C ha R=1: reset a 0 e spostamento in fondo alla lista,
- D ha R=0: swap su disco

Situazione finale in figura.

Pagina	R
E	0
F	1
G	1
A	0
B	0
C	0

5 Gestione della memoria, paginazione e gestione del disco

5.1 Esercizio

Nell'ambito della gestione della memoria con liste concatenate, considerando una lista parzialmente piena di questo tipo (per ogni tripla, il primo elemento è un flag per capire se si tratta di un buco o un processo, il secondo è l'indirizzo di partenza e il terzo è la lunghezza dell'elemento): P,0,6, → H,6,3, → P,9,8 → P,17,4, →H,21,2, → P,23,6, → H,29,4,X.
Dove viene posizionato il nuovo processo P che occupa 4 blocchi? Dove verrebbe posizionato se invece ne occupasse solo 2, secondo gli algoritmi first fit e best fit?

Soluzione

- Lettera **P** rappresenta un processo.
- Lettera **H** rappresenta un "buco" di un certo numero di blocchi.
- **First-fit:** inserisce il nuovo processo nella prima posizione possibile dall'inizio della lista.
- Best-fit: inserisce il nuovo processo nella posizione tale da contenerlo nel modo migliore e, dunque, nel buco più piccolo disponibile per contenere il processo.

Pertanto possiamo dire: **Caso "base", con processo di 4 blocchi**

- P,0,6,→H,6,3,→P,9,8→P,17,4,→H,21,2,→P,23,6,→H,29,4,X

- Spazio necessario per contenere un processo di dimensione 4 blocchi solo a partire dall'indirizzo 29.

Caso first fit, con processo di 2 blocchi

- P,0,6,→H,6,3,→P,9,8→P,17,4,→H,21,2,→P,23,6,→H,29,4,X

Primo spazio sufficiente per contenere il processo inizia all'inidirizzo 6. Si ottiene:

- P,0,6,→P,6,2,→H,8,1,→ P,9,8→P,17,4,→H,21,2,→P,23,6,→H,29,4,X

Caso first fit, con processo di 2 blocchi

- P,0,6,→H,6,3,→P,9,8→P,17,4,→H,21,2,→P,23,6,→H,29,4,X

Miglior spazio per contenere il processo inizia all'inidirizzo 21. Si ottiene:

- P,0,6,→H,6,3,→P,9,8→P,17,4,→P,21,2,→P,23,6,→H,29,4,X

5.2 Esercizio

Utilizzando l'algoritmo di paginazione Aging (con contatore a 8 bit), si considerino le seguenti 4 pagine, con valore di bit R, rispettivamente, 1000. Ai cicli successivi, i valori sono 0111, 0011, 1101, 1100, 0001, 1010 e 1110. Si forniscano i valori dei 4 contatori dopo l'ultimo intervallo, specificando quale pagina viene spostata su disco.

Soluzione

Algoritmo Aging:

- Tabella con andamento dei bit R per ogni pagina.
- Ad ogni nuova scrittura, shift verso destra.
- Swap di pagina con valore binario minore nella riga.

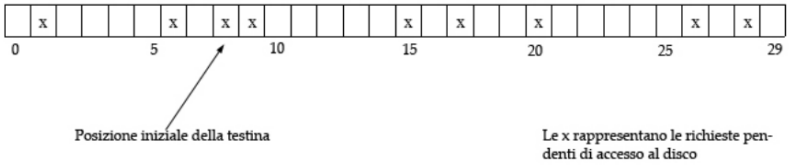
Algoritmo Aging soluzione:

Pagina								
P_0	1	1	0	1	1	0	0	1
P_1	1	0	0	1	1	0	1	0
P_2	1	1	0	0	0	1	1	0
P_3	0	0	1	0	1	1	1	0

Il valore minore binario è P_3.
P_3 viene spostata sul disco.

5.3 Esercizio

Nell'ambito della schedulazione delle richieste al disco, si consideri la situazione delle richieste pendenti e della testina illustrata in figura. Come si comporterà la testina nel caso in cui l'algoritmo di schedulazione utilizzato sarà l'SSF (Shortest Seek First)? Come si comporterà nel caso in cui sarà l'algoritmo dell'ascensore?



Soluzione

(Teoria : I moderni sistemi operativi,da pag 296, paragrafo 5.4.3. Gli algoritmi di schedulazione del braccio del disco.)

Algoritmo SSF (Shortest Seek First): risolte per prima le richieste più vicine alla posizione attuale della testina. La posizione successiva k è stabilita dalla formula seguente:

$$arg\,min_k|(P - P_k)|$$

Algoritmo dell'ascensore: risolte tutte le richieste in una direzione (salita, ad esempio), poi risolte quelle nell'altra direzione.
Garantisce la fairness del sistema, rendendo impossibile la starvation.

Algoritmo SSF soluzione:

8	9	6	1	15	17	20	26	28

Algoritmo dell'ascensore soluzione:

8	9	15	17	20	26	28	6	1