Sistemi Embedded e Real-time (M. Cesati)

Compito scritto del 17 settembre 2010

Esercizio 1. Si consideri il seguente sistema di task periodici in fase schedulato su un processore con un algoritmo "cyclic schedule": $T_1 = (10, 1, 12)$, $T_2 = (15, 1.5, 11)$, $T_3 = (21, 12, 25)$, $T_4 = (32, 4, 24)$. I job sono interrompibili, ma il numero di interruzioni dovrebbe essere ridotto al minimo. Determinare la dimensione del frame che minimizza l'overhead dello scheduler.

Esercizio 2. Si consideri un sistema di tre task periodici con scadenze uguali al periodo così caratterizzati:

	T_1	T_2	T_3	
p_i	3	5	7	(periodo)
e_i	1	1	1	(tempo d'esecuzione)
K_i	1	0	1	(numero di auto-sospensioni)
x_i	0.1	0	0.1	(tempo max auto-sospensione)
θ_i	0.1	0.2	0.1	(tempo max non interrompibilità)

I task T_1 e T_3 accedono alla stessa risorsa condivisa R, per un periodo di tempo massimo rispettivamente pari a 0.1 e 0.2. Il task T_2 non fa uso di risorse condivise. Viene utilizzato un algoritmo "priority-inheritance" per regolare gli accessi alla risorsa condivisa.

(a) Supponendo che venga utilizzato uno scheduler RM con cambio di contesto di costo $CS = 0.05$, determinare analiticamente se il sistema è schedulabile su un singolo processore.

(b) Ripetere l'analisi precedente supponendo che lo scheduler RM sia invocato periodicamente con periodo $p_0 = 0.16$, che il suo overhead sia pari a $e_0 = 0.01$ per ogni invocazione, e che per rendere eseguibile un job si impieghi un tempo massimo pari a $CS_0 = 0.02$.

Esercizio 3. In fase di progettazione di un sistema real-time multiprocessore si prevede di dover eseguire al massimo 25 task periodici $T_i = (2, 1/4)$ e 45 task periodici $T'_i = (3, 1/5)$.

(a) Supponendo di utilizzare uno scheduler partizionato RM-FF, determinare un numero minimo di processori che garantisca il rispetto di tutte le scadenze dei vari task.

(b) Ripetere l'analisi del punto (a) supponendo di utilizzare uno scheduler partizionato EDF-FF.

Sistemi Embedded e Real-time (M. Cesati)

Soluzioni del compito scritto del 17 settembre 2010

Esercizio 1. *Si consideri il seguente sistema di task periodici in fase schedulato su un processore con un algoritmo "cyclic schedule":* $T_1 = (10, 1, 12)$, $T_2 = (15, 1.5, 11)$, $T_3 = (21, 12, 25)$, $T_4 = (32, 4, 24)$. *I job sono interrompibili, ma il numero di interruzioni dovrebbe essere ridotto al minimo. Determinare la dimensione del frame che minimizza l'overhead dello scheduler.*

Determiniamo la dimensione del frame f più appropriata:

- Vincolo sulle fasi dei task: poiché tutti i task hanno fase zero, questa condizione non impone alcun vincolo per f.
- Vincolo sui tempi d'esecuzione dei job:

$$f \geq \max\{1, 1.5, 12, 4\} = 12$$

- Vincolo sulla divisibilità della lunghezza dell'iperperiodo ($\text{mcm}\{10, 15, 21, 32\} = 3360$):

$$f \in \{1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 15, 16, 20, 21, 24, \ldots\}$$

- Vincolo sul task T_1 ($2f - \gcd\{10, f\} \leq 12$):

$$\begin{array}{ll} 2 \cdot 12 - \gcd\{10, 12\} = 22 > 12 & \Rightarrow f \neq 12 \\ f > 12 \Rightarrow 2f - \gcd\{10, f\} > 24 - 10 > 12 & \Rightarrow f \leq 12 \end{array}$$

Non esiste alcuna dimensione del frame maggiore o uguale a 12 che soddisfa tutte le condizioni. Siamo costretti a spezzare (almeno) il task T_3. Per determinare il tempo di esecuzione più lungo dei sotto-job è sufficiente considerare i valori $\{6, 7, 8, 10\}$. Infatti il valore $f = 6$ rispetta sicuramente tutti i vincoli sui task, in quanto $2f = 12$ è minore o uguale alle scadenze di tutti i task (12, 11, 25 e 24). Analizziamo gli altri possibili valori:

- Il valore $f = 10$ viola il vincolo del task T_2 ($2f - \gcd\{15, f\} \leq 11$):

$$2 \cdot 10 - \gcd\{15, 10\} = 15 > 11 \qquad \Rightarrow f \neq 10$$

- Il valore $f = 8$ viola il vincolo del task T_1 ($2f - \gcd\{10, f\} \leq 12$):

$$2 \cdot 8 - \gcd\{10, 8\} = 14 > 12 \qquad \Rightarrow f \neq 8$$

- Il valore $f = 7$ viola il vincolo del task T_1 ($2f - \gcd\{10, f\} \leq 12$):

$$2 \cdot 7 - \gcd\{10, 7\} = 13 > 12 \qquad \Rightarrow f \neq 7$$

In conclusione, il più grande valore ammissibile per la dimensione del frame è $f = 6$. È necessario però spezzare i job di T_3 in sotto-job di durata inferiore a 6.

Esercizio 2. *Si consideri un sistema di tre task periodici con scadenze uguali al periodo così caratterizzati:*

	T_1	T_2	T_3	
p_i	3	5	7	*(periodo)*
e_i	1	1	1	*(tempo d'esecuzione)*
K_i	1	0	1	*(numero di auto-sospensioni)*
x_i	0.1	0	0.1	*(tempo max auto-sospensione)*
θ_i	0.1	0.2	0.1	*(tempo max non interrompibilità)*

I task T_1 e T_3 accedono alla stessa risorsa condivisa R, per un periodo di tempo massimo rispettivamente pari a 0.1 e 0.2. Il task T_2 non fa uso di risorse condivise. Viene utilizzato un algoritmo "priority-inheritance" per regolare gli accessi alla risorsa condivisa.

(a) *Supponendo che venga utilizzato uno scheduler RM con cambio di contesto di costo $CS = 0.05$, determinare analiticamente se il sistema è schedulabile su un singolo processore.*

Osserviamo preliminarmente che le scadenze relative dei task coincidono con i rispettivi periodi. Possiamo dunque cercare di applicare una condizione di schedulabilità basata sul fattore di utilizzazione del sistema di task.

Si osserva che il task T_3 può bloccare direttamente T_1 per 0.2 unità di tempo; inoltre T_3 può bloccare indirettamente anche T_2 per 0.2 unità di tempo a causa del protocollo "priority inheritance".

Per tenere in considerazione l'overhead dello scheduler e dei cambi di contesto, aumentiamo i tempi di esecuzione dei job:

- $e'_1 = e_1 + 4\,(K_1 + 1)\,CS = 1.4$ (accede a risorse condivise)
- $e'_2 = e_2 + 2\,(K_2 + 1)\,CS = 1.1$ (non accede a risorse condivise)
- $e'_3 = e_3 + 4\,(K_3 + 1)\,CS = 1.4$ (accede a risorse condivise)

Siamo dunque in grado di determinare i tempi massimi di blocco di ciascun task:

	T_1	T_2	T_3
$b_i(ss) = x_i + \sum_{k=1}^{i-1} \min\{e'_k, x_k\}$	0.1	0.1	0.2
$b_i(np) = \max_{i<k\leq 3}\{\theta_k\}$	0.2	0.1	0.0
$b_i(rc) = \max_{1\leq k\leq 3}\{B_d(i,k), B_i(i,k)\}$	0.2	0.2	0.0
$b_i = b_i(ss) + (K_i + 1)\,(b_i(np) + b_i(rc))$	0.9	0.4	0.2

Verifichiamo la condizione di schedulabilità per i tre job:

- Schedulabilità di T_1: $\dfrac{e'_1 + b_1}{p_1} = \dfrac{23}{30} < 1 = U_{RM}(1) \qquad \Longrightarrow$ OK
- Schedulabilità di T_2: $\dfrac{e'_1}{p_1} + \dfrac{e'_2 + b_2}{p_2} = \dfrac{23}{30} < 0.8 < U_{RM}(2) \qquad \Longrightarrow$ OK
- Schedulabilità di T_3: $\dfrac{e'_1}{p_1} + \dfrac{e'_2}{p_2} + \dfrac{e'_3 + b_3}{p_3} = \dfrac{961}{1050} > 0.79 > U_{RM}(3) \qquad \Longrightarrow$ no

Applichiamo il test di schedulabilità al task T_3, l'unico che non rispetta la condizione di schedulabilità di Liu-Layland. La funzione di tempo necessario per T_3 è:

$$w_3(t) = e'_3 + b_3 + \lceil t/p_1 \rceil e'_1 + \lceil t/p_2 \rceil e'_2 = 1.6 + \lceil t/3 \rceil \cdot 1.4 + \lceil t/5 \rceil \cdot 1.1$$

Cerchiamo il punto fisso dell'equazione iterativa $w_3(t) = t$:

$$\begin{aligned}
w_3(1.4) &= 1.6 + \lceil 1.4/3 \rceil\, 1.4 + \lceil 1.4/5 \rceil\, 1.1 &&= 4.1\\
w_3(4.1) &= 1.6 + \lceil 4.1/3 \rceil\, 1.4 + \lceil 4.1/5 \rceil\, 1.1 &&= 5.5\\
w_3(5.5) &= 1.6 + \lceil 5.5/3 \rceil\, 1.4 + \lceil 5.5/5 \rceil\, 1.1 &&= 6.6\\
w_3(6.6) &= 1.6 + \lceil 6.6/3 \rceil\, 1.4 + \lceil 6.6/5 \rceil\, 1.1 &&= 8.0\\
w_3(8.0) &= 1.6 + \lceil 8.0/3 \rceil\, 1.4 + \lceil 8.0/5 \rceil\, 1.1 &&= 8.0
\end{aligned}$$

Poiché l'istante 8.0 è oltre la scadenza assoluta del job (7), il task T_3 non è schedulabile.

(b) *Ripetere l'analisi precedente supponendo che lo scheduler RM sia invocato periodicamente con periodo $p_0 = 0.16$, che il suo overhead sia pari a $e_0 = 0.01$ per ogni invocazione, e che per rendere eseguibile un job si impieghi un tempo massimo pari a $CS_0 = 0.02$.*

I tempi di esecuzione e di blocco dei task devono essere aumentati per tenere in considerazione il fatto che lo scheduler è eseguito periodicamente. In particolare:

	T_1	T_2	T_3
$e_i'' = e_i' + (K_i + 1)\, CS_0$	1.44	1.12	1.44
$b_i(np)' = \left(\left\lceil \max_{i<k\leq 3} \theta_k/p_0 \right\rceil + 1\right) p_0$	0.48	0.32	0.16
$b_i' = b_i(ss) + (K_i + 1)\,(b_i(np)' + b_i(rc))$	1.46	0.62	0.52

- Schedulabilità di T_1:

$$\frac{e_0}{p_0} + \frac{e_1'' + b_1'}{p_1} = \frac{1}{16} + \frac{29}{30} = \frac{247}{240} > 1 \qquad \Longrightarrow \text{Non fattibile}$$

- Schedulabilità di T_2:

$$\frac{e_0}{p_0} + \frac{e_1''}{p_1} + \frac{e_2'' + b_2'}{p_2} = \frac{1}{16} + \frac{12}{25} + \frac{87}{250} = \frac{1781}{2000} > 0.83 > U_{RM}(2) \qquad \Longrightarrow \text{no}$$

- Schedulabilità di T_3:

$$\frac{e_0}{p_0} + \frac{e_1''}{p_1} + \frac{e_2''}{p_2} + \frac{e_3'' + b_3'}{p_3} = \frac{1}{16} + \frac{12}{25} + \frac{28}{125} + \frac{7}{25} = \frac{2093}{2000} > 1 \qquad \Longrightarrow \text{Non fattibile}$$

Consideriamo il task T_1: poiché la somma del fattore di utilizzazione e del tempo di blocco rapportato al periodo supera l'unità, si può già concludere che è impossibile garantire il rispetto della scadenza di T_1.

Verifichiamo comunque la non schedulabilità di T_1 studiando la sua funzione di tempo necessario. Benché T_1 abbia priorità massima tra i task del sistema dato, per tenere conto dell'overhead dello scheduler periodico dobbiamo aggiungere al sistema tre task di priorità maggiore di T_1: $T_0 = (p_0, e_0)$, $T_{0,2} = (p_2, CS_0)$ e $T_{0,3} = (p_3, CS_0)$. La funzione di tempo necessario di T_1 diventa perciò:

$$\begin{aligned} w_1(t) &= e_1'' + b_1' + \lceil t/p_0 \rceil\ e_0 + \lceil t/p_2 \rceil\ CS_0 + \lceil t/p_3 \rceil\ CS_0 \\ &= 2.9 + \lceil t/0.16 \rceil\ 0.01 + \lceil t/5 \rceil\ 0.02 + \lceil t/7 \rceil\ 0.02 \end{aligned}$$

Cerchiamo il punto fisso dell'equazione iterativa $w_1(t) = t$:

$$\begin{aligned} w_1(1.44) &= 2.9 + \lceil 1.44/0.16 \rceil\ 0.01 + \lceil 1.44/5 \rceil\ 0.02 + \lceil 1.44/7 \rceil\ 0.02 = 3.03 \\ w_1(3.03) &= 2.9 + \lceil 3.03/0.16 \rceil\ 0.01 + \lceil 3.03/5 \rceil\ 0.02 + \lceil 3.03/7 \rceil\ 0.02 = 3.13 \\ w_1(3.13) &= 2.9 + \lceil 3.13/0.16 \rceil\ 0.01 + \lceil 3.13/5 \rceil\ 0.02 + \lceil 3.13/7 \rceil\ 0.02 = 3.14 \\ w_1(3.14) &= 2.9 + \lceil 3.14/0.16 \rceil\ 0.01 + \lceil 3.14/5 \rceil\ 0.02 + \lceil 3.14/7 \rceil\ 0.02 = 3.14 \end{aligned}$$

Nel caso peggiore un job del task T_1 si conclude 0.14 unità di tempo oltre la propria scadenza assoluta, quindi il sistema non è fattibile.

***Esercizio 3**. In fase di progettazione di un sistema real-time multiprocessore si prevede di dover eseguire al massimo 25 task periodici $T_i = (2, 1/4)$ e 45 task periodici $T'_i = (3, 1/5)$.*

***(a)** Supponendo di utilizzare uno scheduler partizionato RM-FF, determinare un numero minimo di processori che garantisca il rispetto di tutte le scadenze dei vari task.*

Il fattore di utilizzazione dell'algoritmo di schedulazione partizionato RM-FF è

$$U_{RM\text{-}FF}(m) = m \cdot \left(\sqrt{2} - 1\right)$$

ove m è il numero di processori nel sistema. I task periodici da eseguire hanno un fattore di utilizzazione pari a

$$U_T = 25 \cdot \frac{1/4}{2} + 45 \cdot \frac{1/5}{3} = 6 + \frac{1}{8} = 6.125$$

Se $U_T \leq U_{RM\text{-}FF}(m)$, tutte le scadenze dei task sono rispettate, il che significa che m processori sono sufficienti. Si ottiene perciò:

$$m \geq \frac{6.125}{\sqrt{2} - 1} \approx 14.8$$

È quindi necessario includere nel sistema almeno 15 processori.

***(b)** Ripetere l'analisi del punto (a) supponendo di utilizzare uno scheduler partizionato EDF-FF.*

Il fattore di utilizzazione dell'algoritmo di schedulazione partizionato EDF-FF è

$$U_{EDF\text{-}FF}(m, \beta) = \frac{\beta \cdot m + 1}{\beta + 1}$$

ove m è il numero di processori e $\beta = \lfloor 1/\max_k\{e_k/p_k\} \rfloor$.

Nel caso in esame, il massimo fattore di utilizzazione dei task è 1/8, quindi $\beta = 8$. Di conseguenza, un numero sufficiente di processori per garantire la fattibilità del sistema di task si ottiene risolvendo $U_T \leq U_{EDF\text{-}FF}(m, 8)$, ossia:

$$6.125 \leq \frac{8\,m + 1}{8 + 1}$$

che equivale a

$$m \geq \frac{9 \cdot 6.125 - 1}{8} \approx 6.8$$

È quindi necessario includere nel sistema almeno 7 processori.