Report intermedio del progetto "analisi di schedulabilità di un task-set composto da task hard real-time e soft real-time"

Francesco Scandiffio
francesco.scandiffio@stud.unifi.it

I Modello di Task Set

Consideriamo un task-set composto da task hard real-time e task soft real-time eseguiti su un sistema a singolo processore secondo due diverse politiche di scheduling. I task hard real-time sono schedulati secondo una qualche politica che ne garantisce il rispetto delle deadline: non è il focus di questo lavoro verificare che le deadline dei task hard real-time siano rispettate. I task hard real-time sono task periodici, con jitter, offset e tempi di esecuzione dei chunk caratterizzati da una distribuzione di probabilità. Possono condividere risorse che ne richiedano la sincronizzazione per mezzo semafori e mailbox (tipicamente con invio asincrono e ricezione sincrona). Un task soft real-time è caratterizzato da arrivi di Poisson con tasso λ (i.e., il tempo fra l'arrivo di due job consecutivi è una variabile aleatoria avente distribuzione esponenziale con tasso) e da tempi di esecuzione aventi distribuzione esponenziale con tasso . Come estensione, si possono considerare tempi fra due arrivi e tempi di esecuzione con distribuzione iper-esponenziale, esponenziale, o pari alla somma di una distribuzione esponenziale e di una distribuzione Erlang, con l'intento di approssimare la media e il coefficiente di variazione di dati osservati, seguendo l'estensione degli approssimanti di Whitt riportata nel seguente lavoro (vedi sezione III-B): https://stlab.dinfo.unifi.it/carnevali/papers/19 BCPPV THMS.pdf. Un task soft real-time ha una priorità (più alto è il valore, più basso è il livello di priorità). I task soft real-time sono schedulati secondo una modalità best effort basata sul livello di priorità dei task stessi: i task hard real-time hanno precedenza su tutti i task soft real-time, il task

soft real-time a priorità più alta ha precedenza su tutti gli altri task soft real-time, e cosı via... Per ogni task soft real-time, assumiamo un numero massimo di job in coda, una volta raggiunto il quale i job successivi sono scartati. Nota che, dato che i task soft real-time hanno una distribuzione esponenziale del tempo di esecuzione, scartare il nuovo job appena arrivato oppure scartare quello che è attualmente in esecuzione non fa differenza (il remaining time di una variabile aleatoria avente distribuzione esponenziale è una variabile aleatoria avente distribuzione esponenziale con lo stesso tasso). Nota che, agendo sul tasso di arrivo λ, un task soft real-time pu anche essere in qualche modo assimilato ad un insieme di task soft real-time, aventi tutti uno stesso tipo.

II Misure di interesse

II.I Probabilità che il processore non sia in uso ai task hard real-time

Tecnica di soluzione
La probabilità $\alpha(t)$ che nessun task hard real-time stia usando il processore al tempo t può essere calcolata come un semplice reward transiente che il place relativo alla cpu contenga un gettone al tempo t.

Implementazione
La rete di petri è definita in modo hard coded nel codice. Le transizioni dei task hard real-time sono stocastiche con distribuzione uniforme, così da rendere immediata la verifica sul rispetto delle deadline. Il fatto che questi task soddisfino la deadline è una pre-condizione al resto dell'analisi e con distribuzioni uniformi è sufficiente sommare gli LFT delle transizioni e verificare che tale somma sia inferiore al periodo dei task.

Modifiche
Vorrei costruire la rete di petri in modo dinamico, dando in input una lista di task hard real-time. Inoltre vorrei non limitarmi a considerare distribuzioni di probabilità uniformi.

II.II Probabilità di avere n job di un task soft real-time in coda al tempo t

La probabilità $Q(t)$ di avere n job del task soft real-time a priorità più alta in coda al tempo t (per $t \in [0, T]$ dove T è l'iper periodo dei task hard real-time) si può calcolare risolvendo il seguente sistema di equazioni differenziali dove $R_{k,j}(t) := P\{Q(t) = j | Q(0) = k\}$. (Fig 1)

$$
\begin{cases}
R'_{k0}(t) &= \mu f_\mu(1)\alpha(t)R_{k1}(t) \\
&\quad -\lambda f_\lambda(0)R_{k0}(t) \\
R'_{kj}(t) &= -R_{kj}(t)\,\alpha(t)\mu f_\mu(n) \\
&\quad -R_{kj}(t)\lambda f_\lambda(n) \\
&\quad +R_{k,j+1}(t)\,\alpha(t)\mu f_\mu(n+1) \\
&\quad +R_{k,n-1}(t)\lambda f_\lambda(n-1) \\
R'_{k,K}(t) &= R_{k,K}(t)\,\alpha(t)\mu f_\mu(n) \\
&\quad +R_{k,K-1}(t)\lambda f_\lambda(n-1)
\end{cases}
$$

Figure 1: Equazioni differenziali per il calcolo della probabilità di avere n job di task soft real-time in coda al tempo t.

Implementazione
Il sistema di equazioni differenziali è stato tradotto in codice e viene usato nel caso in cui la distribuzione sia di tipo esponenziale. Il sistema è stato anche esteso per poter rimanere valido in caso di distribuzioni IperEsponenziale, Erlang, somma di Esponenziale ed Erlang.
In ogni caso i tassi λ e μ sono moltiplicati ciascuno per una funzione del numero di job in coda: per semplicità assumiamo che queste funzioni ritornino sempre 1 e per questo motivo sono state omesse dall'implementazione.

Modifiche
Potrei aggiungere nuovamente le funzioni $f_\lambda(x)$ ed $f_\mu(x)$ e permettere all'utente di specificare la formula da utilizzare per il calcolo, usando $f(x) = 1 \; \forall \; x$ come valore di default.

II.III Probabilità di avere n job di un task soft real-time in coda al tempo t

Questo punto è molto simile al precedente dal punto di vista implementativo. La difficoltà maggiore era relativa all'estensione del sistema di equazioni differenziali ma ricordo di averlo controllato varie volte in passato, anche insieme a lei, quindi non dovrebbero esserci problemi su questo.

II.IV Distribuzione steady-state del numero di job in coda del task soft real-time a priorità pù alta all'inizio di ogni iper-periodo dei task hard real-time

Tecnica di soluzione
Calcoliamo la distribuzione steady-state solo sul task soft real-time a priorità maggiore perché è l'unico che risente di un *disturbo periodico* nella sua esecuzione. Per i task a priorità inferiore non si raggiunge mai uno steady state.

Implementazione
Niente di particolare da segnalare a livello implementativo, magari da ricontrollare a livello di formula. Ho usato la seguente:

$$\underline{\pi}(m+1) = \underline{\underline{P}} \; \underline{\pi}(m) \; \forall m \in \{0, 1, ..., n\} \tag{1}$$

con $\underline{\underline{P}} := [P_{k,j}]$

II.V Probabilità di deadline miss

Come estensione avevamo detto di calcolare anche la probabilità che i task soft real-time mancassero una o più volte la propria deadline. Per semplicità l'analisi si limita ad un solo task soft real-time.

Tecnica di soluzione
Consideriamo una deadline relativa D e supponiamo che all'istante t_0 il $k-esimo$ job sia aggiunto all'elenco dei job in coda per il task soft real-time in esame. Il job k manca la deadline se non è eseguito entro il tempo $t_0 + D$. La probabilità che questo evento avvenga è equivalente a

$$\varphi_{k,t_0} := 1 - Prob\{\texttt{K job sono eseguiti entro l'istante } t_0 + D\}$$

Poiché l'unica risorsa necessaria per eseguire questi job è il processore, allora il calcolo della probabilità che i job siano eseguiti può essere semplificato come il calcolo della probabilità che il processore sia disponibile per un tempo sufficiente a completare tutti i job in coda nell'intervallo $[t_0, t_0 + D]$.
Eseguendo un'analisi transiente è possibile ottenere la CDF del tempo di esecuzione di 1 job. Applicando la derivata si ottiene la pdf per un singolo job. La pdf del job k è il risultato di una serie di convoluzioni di pdf successive:

$$pdf_i = pdf_{i-1} \bigotimes pdf_1$$

- Ricordo che in passato c'erano stati dei dubbi su questo punto delle convoluzioni successive, quindi potrebbe essere utile ripercorrere questo ragionamento durante il prossimo ricevimento. -
Integrando la pdf_k si ottiene la cdf_k con cui calcolare la probabilità φ. In conclusione, la probabilità obiettivo si ottiene risolvendo

$$\sum_{k=1}^{k=N} \varphi_{k,t_0} \cdot Q_{k-1}(t)$$

Implementazione
Le derivate le ho implementate usando il metodo della differenza finita centrata

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{2}$$

Avevo provato a confrontare i risultati del mio programma con risultati ottenuti su geogebra ma dalla terza convoluzione in poi compariva un errore relativo del 5%.

III Esperimenti

Effettuati ma da inserire in relazione e discutere

1. Fisso dei parametri di un insieme di task hard real-time. Considero un solo task soft real-time, facendo variare le distribuzioni degli arrivi e del tempo di servizio. La condizione di partenza della coda la tengo invariata tra i vari esperimenti (direi sempre vuota). Al variare delle distribuzioni calcolo le misure di interesse indicate sopra e faccio dei confronti.

Programmati

1. Esperimento con carico di lavoro variabile.
   Dato un task soft real-time con coda vuota e le distribuzioni A per gli arrivi ed S per il servizio, calcolo lo steady state a partire dalla coda vuota e osservo quanto ci mette il transitorio a convergere allo steady state che ho calcolato, con un certo margine di errore. Ripeto il procedimento ma con le seguenti modifiche:
   - modifico solo la distribuzione degli arrivi, diciamo in A'
   - al posto di partire dalla coda vuota uso come condizione di partenza lo steady state del punto precedente.
2. Oltre che un solo task soft real-time potrei considerarne due e valutare le misure transitorie su quello a priorità inferiore. Potrei tenere le distribuzioni di questo secondo task fisse, mostrando come lo stato della sua coda sia influenzato da diverse combinazioni di distribuzioni del task soft real-time a priorità maggiore.
3. Esperimento sulla non scalabilità del sistema.
   Concettualmente è chiaro che non sarà possibile analizzare una rete composta da troppi task a causa dell'esplosione degli stati, potrebbe essere interessante mostrare quanto ci vuole per raggiungere questo limite. Per esempio anche a parità di task, l'analisi con distribuzione come somma di Esponenziali ed Erlang avrà più stati rispetto a quella con con sole Esponenziali, con conseguente aumento della complessità computazionale.