



Lezione R12

Real-time su multiprocessore II

Sistemi embedded e real-time

22 dicembre 2022

Marco Cesati

Dipartimento di Ingegneria Civile e Ingegneria Informatica
Università degli Studi di Roma Tor Vergata

SERT22

R12.1



Di cosa parliamo in questa lezione?

In questa lezione continuiamo a parlare del problema della schedulazione real-time in sistemi multiprocessore, con particolare enfasi sugli scheduler globali

- ① EDF-FFDD
- ② Scheduler globali
- ③ EDF globale
- ④ EDF-US[ζ]
- ⑤ EDF(k)
- ⑥ RM globale
- ⑦ RM-US[ζ]

Algoritmo EDF-FF

Real-time su multiprocessore II

Marco Cesati



L'euristica “first fit” accoppiata all'algoritmo di schedulazione **EDF** dà luogo all'algoritmo di allocazione “on-line” **EDF-FF**:

- ➊ ordina arbitrariamente i processori: P_1, P_2, \dots
- ➋ assegna ciascun task T_i al primo processore P_j tale che l'insieme dei task già assegnati a P_j insieme a T_i risulta ancora schedulabile tramite **EDF**

- $U_{\text{EDF-FF}} = \frac{\beta \cdot m + 1}{\beta + 1}, \quad \beta = \left\lfloor 1 / \max_k \frac{e_k}{p_k} \right\rfloor$ (Lopez & al., 2000)
- Fattore di approssimazione: 1.7 (Garey & Johnson, 1979)

EDF-FF è ottimale tra tutti gli algoritmi partizionati:

$$\beta = 1 \implies U_{\text{EDF-FF}} = (m+1)/2$$

$$\beta \rightarrow \infty \implies U_{\text{EDF-FF}} \rightarrow m$$

SERT22

R12.3

Algoritmo EDF-FFDD

Real-time su multiprocessore II

Marco Cesati



È possibile estendere gli algoritmi partizionati basati su “first fit” anche a task sporadici con **scadenze arbitrarie**

Ad esempio, **EDF-FFDD** (Baruah & Fisher 2005):

- ➊ ordina arbitrariamente i processori: P_1, P_2, \dots
- ➋ ordina i task sporadici per densità $\frac{e_i}{\min\{p_i, d_i\}}$ decrescenti
- ➌ assegna ciascun task T_i al primo processore P_j tale che l'insieme dei task già assegnati a P_j insieme a T_i **risulti ancora schedulabile tramite EDF**

Condizione di schedulabilità: m è n° processori, lo fisso (non aumentano magicamente)

$$\Delta_T \leq \begin{cases} m - (m-1)\Delta_{\max} & \text{se } \Delta_{\max} \geq 1/2 \\ m/2 + \Delta_{\max} & \text{se } \Delta_{\max} \leq 1/2 \end{cases}$$

SERT22

R12.4



Schema della lezione

EDF-FFDD

Scheduler globali

EDF globale

EDF-US[ζ]EDF(K)

RM globale

RM-US[ζ]

Ha senso considerare algoritmi di schedulazioni multiprocessore globali (non partizionati)?

Assumiamo in questa lezione che tutti i job siano indipendenti, interrompibili, migrabili e senza auto-sospensioni

Problematiche da affrontare:

- anomalie di schedulazione
- utilità rispetto a EDF partizionato
- istanti critici
- effetto Dhall

SERT22

R12.5

Anomalie di schedulazione



Schema della lezione

EDF-FFDD

Scheduler globali

EDF globale

EDF-US[ζ]EDF(K)

RM globale

RM-US[ζ]

I sistemi multiprocessori presentano **anomalie di schedulazione** (cfr. R11.15)

Teorema (Ha & Liu, 1994)

Sistemi di task periodici interrompibili e migrabili su multiprocessore schedulati con algoritmi a priorità fissa (a livello di task o di job) sono predicibili e quindi non presentano anomalie di schedulazione dipendenti dal tempo di esecuzione dei job

Purtroppo esistono altre anomalie legate alla variazione di altri parametri temporali

però quelle peggiori, legate ai tempi di esecuzioni, riesco ad arginarle per il teorema.

SERT22

R12.6

Utilità rispetto a EDF partizionato

Real-time su multiprocessore II

Marco Cesati



Algoritmi a priorità fissa a livello di job come **EDF** sono:

- ottimali nel caso uniprocessore
- tra i migliori possibili negli scheduler multiprocessore partizionati (vedi EDF-FF)

Quale vantaggio avrebbero gli scheduler globali?

Tra gli scheduler **partizionati EDF** è ottimale, ma **non è ottimale in assoluto**. Esistono algoritmi globali a priorità dinamica a livello di job che sono ottimali (cfr. algoritmo **Pfair**)

SERT22

R12.7

Scheduler globali a priorità dinamica a livello di job

Real-time su multiprocessore II

Marco Cesati



Teorema (Baruah 2008)

Un sistema di task sporadici con scadenze arbitrarie è schedulabile con m processori con un **algoritmo globale** a priorità dinamica a livello di job se

$$\Delta_T = \sum_i \frac{e_i}{\min_i\{d_i, p_i\}} \leq m \quad \text{e} \quad \Delta_{\max} = \max_i \frac{e_i}{\min\{p_i, d_i\}} \leq 1$$

Corollario

Un sistema di task sporadici con scadenze implicite è schedulabile con m processori con un **algoritmo globale** a priorità dinamica a livello di job se e solo se

$$U_T = \sum_i \frac{e_i}{p_i} \leq m \quad \text{e} \quad U_{\max} = \max_i \frac{e_i}{p_i} \leq 1$$

SERT22

R12.8

Istanti critici

Real-time su multiprocessore II

Marco Cesati



Il rilascio in fase di tutti i job non corrisponde necessariamente ad un istante critico (cfr. R11.18)

Non è possibile o pratico analizzare il sistema tramite la funzione di tempo necessario: come dimostrare la schedulabilità di un sistema?

Possiamo utilizzare le condizioni di schedulabilità legate al fattore di utilizzazione o alla densità dei task!

SERT22

R12.9

Effetto Dhall

Real-time su multiprocessore II

Marco Cesati



Vale comunque l'**effetto Dhall**: dato un qualunque numero $m > 1$ di processori, esistono sempre insiemi di task periodici schedulabili con utilizzazione totale costante (rispetto a m) che non possono essere schedulati con un algoritmo RM, DM o EDF (cfr. R11.12,13)

L'**effetto Dhall** è condizionato alla presenza di un task periodico con utilizzazione vicina all'unità. Possiamo quindi correlare l'utilizzazione massima dei vari task con la schedulabilità del sistema

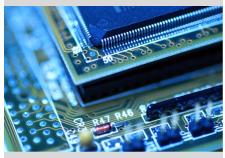
SERT22

R12.10

Condizione di schedulabilità per EDF globale

Real-time su multiprocessore II

Marco Cesati



Teorema (Srinivasan & Baruah 2002; Goossens, Funk & Baruah 2003)

Un sistema T di task sporadici con scadenze implicite, utilizzazione totale U_T , e utilizzazione massima dei task U_{\max} , è schedulabile con EDF globale su un sistema con m processori se

$$U_T \leq m - (m - 1) \cdot U_{\max}$$

Casi limite:

- $U_{\max} = 1 \Rightarrow U_T \leq 1 \Rightarrow$ effetto Dhall
- $U_{\max} = 0 \Rightarrow U_T \leq m \Rightarrow$ EDF globale è ottimale

Il teorema vale per scadenze non implicite?

Questa funzione di schedulabilità non parla di densità, né di scadenze (che possono essere più piccoli dei periodi) allora questo risultato NON VALE.

SERT22

R12.11

Esiste però un'estensione di questo teorema per sistemi di task a scadenza arbitraria:

Condizione di schedulabilità per EDF globale (2)

Real-time su multiprocessore II

Marco Cesati



Teorema (Bertogna, Cirenei & Lipari 2005)

Un sistema T di task sporadici con scadenze arbitrarie, densità totale Δ_T , e densità massima dei task Δ_{\max} , è schedulabile con EDF globale su un sistema con m processori se

$$\Delta_T \leq m - (m - 1) \cdot \Delta_{\max}$$

Esistono diverse altre condizioni di schedulabilità per EDF globale basate sulla densità dei singoli task e/o sul calcolo del tempo di processore richiesto dai task

È possibile avere algoritmi di schedulazione a priorità fissa migliori di EDF globale per insiemi di task generici?

Abbiamo visto che quando ci sono task molto grandi, allora ho effetto Dhall che crea problemi per algoritmi GLOBALI (inefficienti). Ciò non sembra affliggere invece i PARTIZIONATI, che soffrono invece task molto piccoli, i quali non danno problemi ai globali.

SERT22

R12.12



Schema della lezione

EDF-FFDD

Scheduler globali

EDF globale

EDF-US[ζ]

EDF(k)

RM globale

RM-US[ζ]

Algoritmo EDF-US[ζ]

Real-time su multiprocessore II

Marco Cesati

EDF-US[ζ] è un algoritmo di **schedulazione “ibrido”**

- Proposto da Srinivasan e Baruah nel 2002
- ζ è un parametro dell’algoritmo, $\zeta \leq 1$ (tale simbolo è una Zeta greca)
- Una parte dei task ha **priorità fissa**
 - Sono i task T_i tali che $e_i/p_i > \zeta$
 - Tutti questi hanno **identica priorità**
- L’altra parte dei task ha priorità inferiore
 - Schedulati con **EDF**
- **Idea fondamentale:**
 - ai task “grandi” sono assegnati per primi i processori disponibili
 - i task “piccoli” vengono schedulati con il tempo di processore rimanente



Schema della lezione
EDF-FFDD
Scheduler globali
EDF globale
EDF-US[ζ]
EDF(k)
RM globale
RM-US[ζ]

SERT22 R12.13

Esempio di schedulazione EDF-US[ζ]

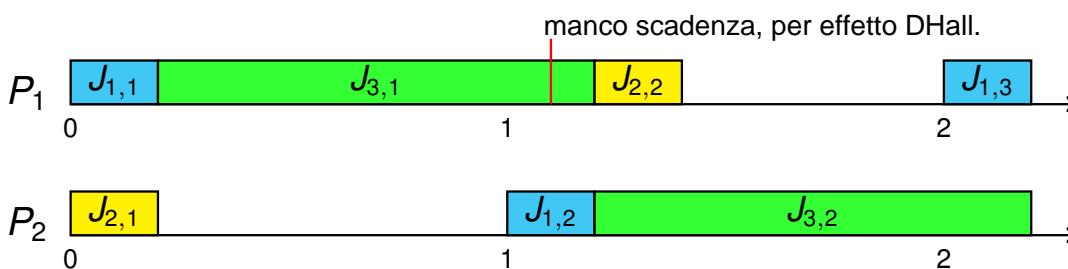
Real-time su multiprocessore II

Marco Cesati

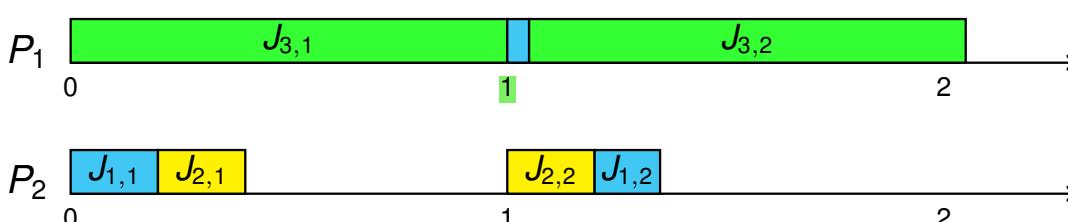
Consideriamo $T_1 = (1, 1/10)$, $T_2 = (1, 1/10)$, $T_3 = (21/20, 1)$
($U_T = 121/105$)

uso due processori, perchè $U_t < 2$

Schedulazione con **EDF globale**:



Schedulazione con **EDF-US[1/2]**:



Schema della lezione
EDF-FFDD
Scheduler globali
EDF globale
EDF-US[ζ]
EDF(k)
RM globale
RM-US[ζ]

SERT22 R12.14



Teorema (Srinivasan & Baruah, 2002)

Un sistema di task sporadici con scadenze implicate è schedulabile con $\text{EDF-US}[m/(2m - 2)]$ su m processori se

$$U_T \leq m^2/(2m - 2) \quad \text{che garanzia mi da..?}$$

Corollario (Baker, 2005)

Un sistema di task sporadici con scadenze implicate è schedulabile con $\text{EDF-US}[1/2]$ su m processori se

questo è il meglio che si può fare.

$$U_T \leq (m + 1)/2$$

Nessun algoritmo di schedulazione globale con priorità fissa a livello di task o job può avere fattore di utilizzazione maggiore di $(m + 1)/2$ (Andersson et. al. 2001, cfr. R11.19)

Possono esistere algoritmi di schedulazione globale a priorità fissa migliori di EDF-US[1/2]?

Si!

- Schema della lezione
- EDF-FFDD
- Scheduler globali
- EDF globale
- EDF-US[ζ]**
- EDF(k)
- RM globale
- RM-US[ζ]

SERT22 R12.15

Algoritmo EDF(k)

EDF(k) è un algoritmo di schedulazione “ibrido”

- Proposto da Goossens, Funk e Baruah nel 2003
- k è un parametro dell’algoritmo, $k < m$
- Una parte dei task ha priorità fissa
 - Sono i $k - 1$ task T_i che hanno fattore di utilizzazione e_i/p_i più alto
 - Tutti questi hanno identica priorità
- L’altra parte dei task ha priorità inferiore
 - Schedulati con EDF
- Medesima idea fondamentale: come visto prima.
 - ai task “grandi” sono assegnati per primi i processori disponibili
 - i task “piccoli” vengono schedulati con il tempo di processore rimanente



- Schema della lezione
- EDF-FFDD
- Scheduler globali
- EDF globale
- EDF-US[ζ]
- EDF(k)**
- RM globale
- RM-US[ζ]

SERT22 R12.16

Prestazioni di EDF(k)

Teorema (Goossens, Funk & Baruah, 2003)

Un sistema di task sporadici con scadenze implicite è schedulabile con EDF(k) su m processori se

$$(k - 1) + \left\lceil \frac{U_T - u_k}{1 - u_k} \right\rceil \leq m$$

ove u_k è il fattore di utilizzazione del k -esimo task (ordinando i task per fattore di utilizzazione non crescente)

Algoritmo EDF(k_{\min}) (Baker 2005)

Sia fissato m (il numero di processori), e sia k_{\min} il minimo valore di k che soddisfa la condizione del teorema precedente. Un sistema di task sporadici con scadenze implicite è schedulabile con EDF(k_{\min}) se $U_T \leq (m + 1)/2$

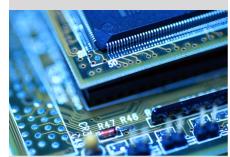
L'insieme di sistemi di task schedulabili da EDF-US[1/2] è un sottoinsieme proprio dell'insieme di EDF(k_{\min})!

Vuol dire che, tutto ciò che è sotto $(m+1)/2$ è garantito essere schedulabile, ma ciò non toglie che ci siano insiemi di task con utilizzazione totale $> (m+1)/2$ che siano comunque schedulabili. EDF(k_{\min}) è meglio perché "ingloba" più insiemi rispetto a EDF-US[1/2], perché riesce a schedulerne di più quando essi eccedono $(m+1)/2$. Se prendo solo $(m+1)/2$ sono uguali ovviamente.

Scheduler globali basati su priorità fissa a livello di task

Scheduler globali o ibridi che utilizzano protocolli a priorità fissa a livello di task

- non possono avere prestazioni superiori agli scheduler globali o ibridi basati su priorità dinamica a livello di task
- in pratica sono molto considerati, perché
 - semplici da implementare
 - presenti in ogni RTOS
 - più robusti in caso di job che superano il WCET calcolato in fase di design





Teorema (Andersson, Baruah & Jonsson 2001)

Un sistema di task periodici con scadenze implicite tale che $U_{\max} \leq m/(3m - 2)$ è schedulabile globalmente con RM su m processori se

$$U_T \leq \frac{m^2}{2}/(3m - 1)$$

Teorema (Baker 2003; Bertogna, Cirenei & Lipari 2005)

Un sistema di task sporadici con scadenze implicite è schedulabile globalmente con RM su m processori se

$$U_T \leq \frac{m}{2} (1 - U_{\max}) + U_{\max}$$

Esistono anche condizioni di schedulabilità per sistemi di task con scadenze arbitrarie ma hanno in genere formulazioni più complicate

Algoritmo RM-US[ζ]



RM-US[ζ] è un algoritmo di schedulazione “ibrido”, analogo a EDF-US[ζ]

- Proposto da Andersson, Baruah e Jonsson nel 2001
- ζ è un parametro dell’algoritmo, $\zeta \leq 1$
- Una parte dei task ha priorità fissa
 - Sono i task T_i tali che $e_i/p_i > \zeta$
 - Tutti questi hanno identica priorità
- L’altra parte dei task ha priorità inferiore
 - Schedulati con RM
- Medesima idea fondamentale di EDF-US[ζ]:
 - ai task “grandi” sono assegnati per primi i processori disponibili
 - i task “piccoli” vengono schedulati con il tempo di processore rimanente

Prestazioni di RM-US[ζ]

Real-time su multiprocessore II

Marco Cesati



Teorema (Andersson, Baruah & Jonsson 2001)

Un sistema di task periodici con scadenze implicite è schedulabile con RM-US[$m/(2m - 2)$] su m processori se

$$U_T \leq m^2/(3m - 2)$$

Teorema (Baker 2003; Bertogna, Cirenei & Lipari 2005)

Un sistema di task sporadici con scadenze implicite è schedulabile con RM-US[1/3] su m processori se

$$U_T \leq (m + 1)/3$$

Il miglior valore possibile per ζ è $\zeta = 0,37482$, con utilizzazione massima pari a $0,37482 \cdot (m + 1)$ (Lundberg 2002)

Sono stati studiati anche gli algoritmi RM(k) e RM(k_{\min}), analoghi a quelli basati su EDF

Schema della lezione
EDF-FFDD
Scheduler globali
EDF globale
EDF-US[ζ]
EDF(k)
RM globale
RM-US[ζ]

SERT22 R12.21

Il grosso problema ancora non risolto quando parlo di sistemi multiprocessore è che tutti gli algoritmi sono basati su ipotesi fondamentali:

indipendenza dei task (non ho mai parlato di risorse condivise o vincoli). Togliendo tali ipotesi perdo tutti i risultati e c'è ben poco di valido. Solo questo? NO.

I vincoli di dipendenza dei task possono essere nascosti: ad esempio, ho diversi core, ogni core ha una cpu, e ho quindi un sistema parallelo multiprocessore in cui il singolo processore è multicore... è un caso abbastanza comune.

Core e processori NON SONO INDIPENDENTI, condividono risorse hw (I core potrebbero non usare singola MMU per accesso alla RAM, oppure singola cache di 2° e 3° livello. I processori possono avere dipendenze per accesso al bus di sistema).

Quindi, anche operando su unico core, ho sempre dipendenza rispetto ad altri core/bus/cache se c'è anche un solo altro job. Non posso parlare di worst case con un solo job, perché vale solo se c'è quell'unico job.

Gli enti certificatori non certificheranno mai questi sistemi, perché non è mai vero che: job indipendenti su processori dipendenti non si influenzano a vicenda, è FALSO.

L'unica cosa da fare, livello hardware, è creare sistemi multiprocessore che si comportano analogamente a sistemi uniprocessore, a preiscindere da quello che fanno gli altri core nel sistema. Questo è un obiettivo non raggiunto. Se risolvessi questo problema, sarei milionario.

Nei sistemi realtime certificati non posso usare multiprocessori, o meglio: devo renderlo singolo processore, cioè nel sistema di task deve esserci un task periodico con priorità massima avente compito di spegnere tutti i processori del sistema tranne uno, cioè ne uso solo uno per ritornare al caso singolo processore. Inoltre periodicamente controlla e rispegne tutti i processori tranne uno, sennò non avrà mai la certificazione.