Assunzioni RUN

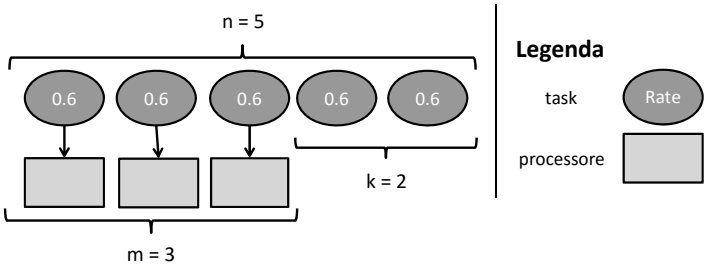
Parametri di modello

- m processori
- *Implicit-deadline task* $\tau_i, i \in \{1..n\}$ con $n = m + k \geq m$
- *Fully utilized system*: non ci devono essere tempi morti
- Ciascun task ha *fixed rate* $R_i = \frac{(d_i - r_i)}{C_i}$,
  - Che esprime la pendenza della fluid curve
- Task indipendenti
- Processori omogenei
- *Migration* e *preemption* sono assunti avere costo zero
- $\sum_{i=1}^{n}(R_i) \leq m$

  ⇒ SI PUO' FARE! Ma come? Quale Σ *scheduling* valido?

1

Esempio



- $\sum_{n=1}^{5}(R_n) = 3$ ⇒ almeno 3 processori
- Voglio trovare lo schedule Σ del **sistema S**

2

Sistema duale

L'algoritmo RUN crea l'**insieme duale** dei *task*

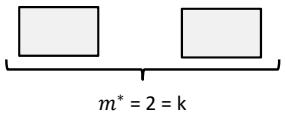
- per ogni *task* τ_i introduciamo τ_i^* che esegue quando τ_i è *idle* e viceversa
- due *schedule* Σ e Σ^* sono duali quando

  $\forall t, \tau_i \in \Sigma(t)$ se e solo se $\tau_i^* \notin \Sigma^*(t)$

Tale insieme di *task* viene eseguito nell'**architettura virtuale** del sistema originale (sistema duale)

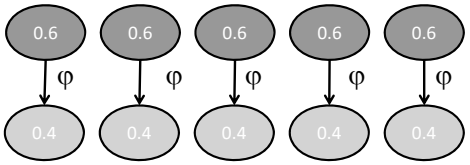
3

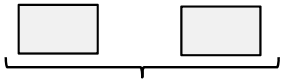
Esempio: S^*



4

Esempio: S^*

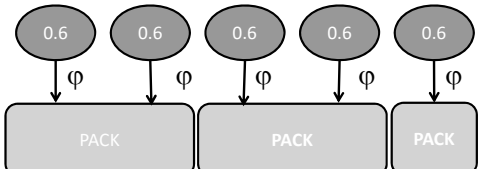




Legenda:
φ = operazione **DUAL**

5

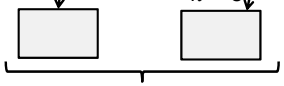
Esempio: S^*



$\psi = \sigma \circ \varphi$

n^* = 3

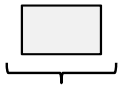
k^* = 3 -2 = 1



Legenda:
φ = operazione **DUAL**
σ = operazione **PACK**

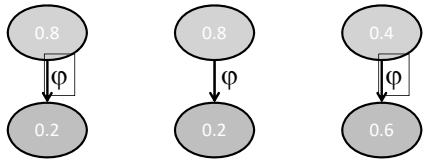
6

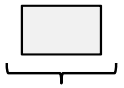
Esempio: S^{**}



7

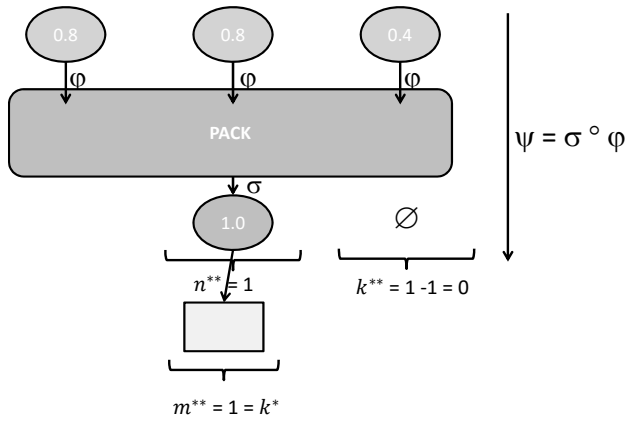
Esempio: S^{**}





8

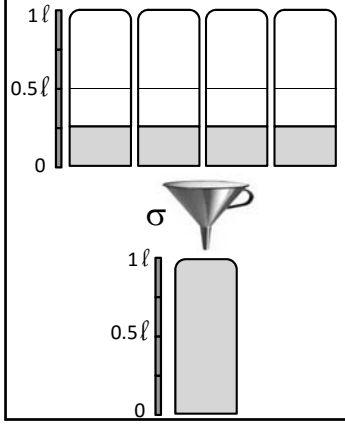
Esempio: S^{**}



9

Algoritmo di riduzione: terminazione

Lemma: $\psi = \left|\sigma \circ \varphi\,(U_1^4 \tau_i)\right| \leq \left\lceil \frac{|\tau|+1}{2} \right\rceil$

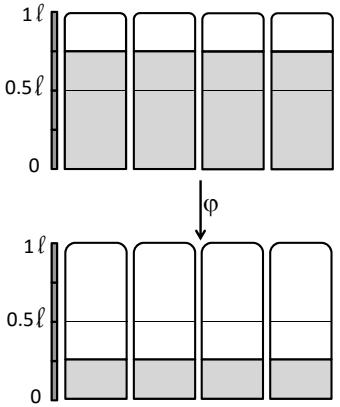


⇒ ψ -REDUCE (σ ° φ) ha termine, perché a ogni passo si riduce il *workload* (*rate* complessivo), e quindi anche il numero di processori necessari, e con σ -PACK almeno dimezzo il numero di task

Teorema terminazione: dopo un numero finito ***p*** di iterazioni l'algoritmo termina riducendo ad un sistema uniprocessore (*rate* complessivo uguale a uno)

Algoritmo di riduzione: terminazione

Lemma: $\psi = \left|\sigma \circ \varphi\,(U_1^4 \tau_i)\right| \leq \left\lceil \frac{|\tau|+1}{2} \right\rceil$



Intuizione

$\sum_1^4 R = 3 \Rightarrow$ m = 3
n = 4
k = 1 < m

Nel sistema duale
$\sum_1^4 R = 1 \Rightarrow m^* = 1 =$ **k**
$n^* = 1$
$k^* = 0$

Passi fondamentali di RUN

Riassumendo:

- Operazione DUAL (**φ**)
- Operazione PACK (**σ**)
- Operazione REDUCE (**ψ** = σ ° φ) riduce la dimensione del problema a ogni passo (~ dimezza)
- Teorema (validità del duale): Σ valido ⇔ Σ^* valido
- Poiché ogni *task* duale rappresenta il tempo *idle* del *task* primario, trovare lo *schedule* del sistema duale è equivalente a trovare quello del sistema primario

Scheduling

L'algoritmo genera un albero dalle foglie alla radice (fase *off-line*)

Lo *schedule* si ricava ripercorrendo a ritroso l'albero generato dall'algoritmo e applicando ad ogni istante *t* le seguenti regole (fase *on-line*)

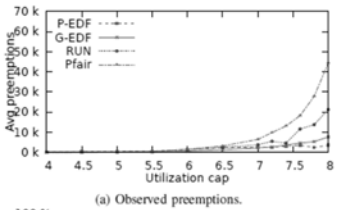
- ai passi di tipo φ si usa la regola del duale (esegui in $\Sigma(t)$ i task che non eseguono in $\Sigma^*(t)$)
- ai passi di tipo σ si sceglie fra i *task* aggregati usando EDF

Il problema multiprocessore originario è stato quindi ricondotto a una collezione di problemi uniprocessore (**composizionalità** del problema)

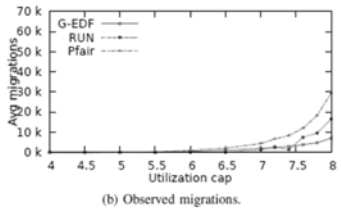
8.d A stint on RUN

Credits to E. Mezzetti and D. Compagnin (ECRTS 2014)

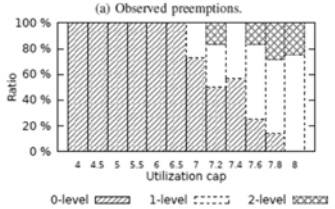
Implementation experience /1



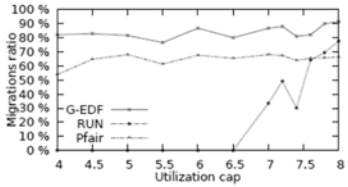
(a) Observed preemptions.



(b) Observed migrations.

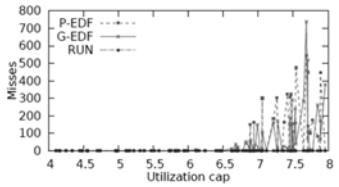


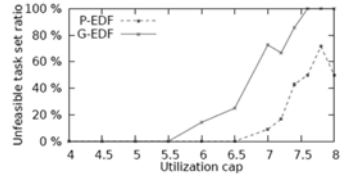
(c) Tree levels per utilization.

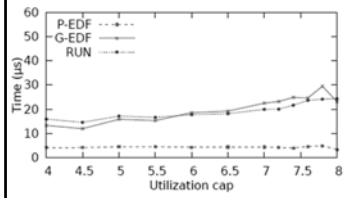


(d) Migration ratio.

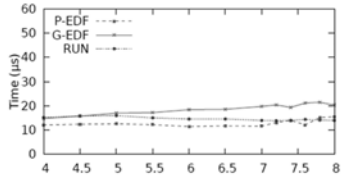
Implementation experience /2







(a) Average release overhead.



(b) Average schedule overhead.