5. Statistisches ratenmonotones Scheduling SRMS

http://www.cs.bu.edu/groups/realtime/SRMSworkbench

5.1. Ausgangspunkt

- **Motivation**

  Schwankende Bearbeitungszeiten

  Betriebsmittel-Auslastung

  Überlast

- **Grundidee**

  Periodische Task muß nur zu bestimmtem Prozentsatz Deadline einhalten, gesteuert durch einen „Kredit“ über einen bestimmten Zeitraum („Superperiode“).

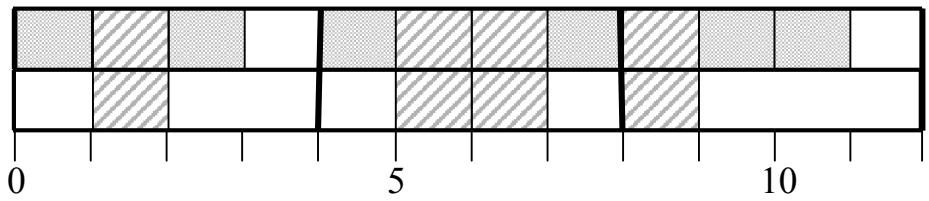
  Dazu muß Bearbeitungszeit am Beginn jeder Periode bekannt sein.

  Kredit: Ergebnis einer „Qualitäts-Aushandlung“

- **Hintergrund**

  Tasktransformation $\tau_i = (t_i, e_i) \mapsto \hat{\tau}_i = (t_{i+1}, \hat{e}_i)$ mit $\hat{e}_i = \frac{t_{i+1}}{t_i} \cdot e_i$

  Beispiel. $t_3 = 4, \quad e_3 = 2, \quad t_4 = 12 \quad \rightarrow \quad \hat{e}_3 = \frac{12}{4} \cdot 2 = 6$



Es gilt: Ist $\{(t_1, e_1), \ldots, (t_i, e_i), (t_{i+1}, e_{i+1}), \ldots\}$ gemäß RMS einplanbar,

so auch $\{(t_1, e_1), \ldots, (t_{i+1}, \hat{e}_i), (t_{i+1}, e_{i+1}), \ldots\}$.

5.2. SRMS-Modell

- **Taskbeschreibung**

$T = \{\tau_1,...,\tau_n\}$ periodische Tasks; n: Anzahl (fest)

$\tau_i = (\tau_{ij})_{j=1,2,...}$ τ_{ij} : j-ter Job von Task τ_i, $i = 1,...,n$

$\tau_i : (t_i, f_i, Q_i)$ t_i: Periode = Deadline $d_{ij} = j \cdot t_i$

f_i: Dichtefunktion der Bearbeitungszeit von τ_{ij}

$f_i \equiv 0$ für $t \notin [0, t_i]$

Q_i: QoS-Parameter; Wkt., daß τ_i Deadline erreicht

e_{ij} Bearbeitungszeit (BM-Bedarf) von τ_{ij}

r_{ij} Bereitzeit (Freigabezeit) von τ_{ij}; $r_{ij} = (j-1)t_i$

o.B.d.A. T geordnet gemäß t_i $(t_1 \le t_2 \le ...)$

- **Superperiode von τ_i**

t_{i+1}

- ***T* einplanbar (feasible)**

Jede Task kann zu Beginn jeder Superperiode ihren Kredit erhalten.

$\rightarrow$ Alle $\tau_i \in T$ erreichen (ausgehandelten) QoS-Parameter.

5.3. Prinzipien und Vorgehen

- **Scheduling**

  ereignisgesteuert mittels fester Prioritäten, preemptiv

- **Admission control**

– ***lokal:*** zur Bereitzeit jedes Jobs.

Grundsätze: kein Job überschreitet seinen BM-Bedarf e_{ij}

kein Job wird zugelassen, wenn Deadline-Einhaltung nicht garantiert werden kann → „Taskisolation“

nicht zugelassene Jobs:

verwerfen | auf aktuell niedrigste Priorität setzen

– ***global:*** Zulassung einer Task

Aushandeln des QoS-Parameterwerts

- **Vorgehen**

– τ_i besitzt „Konto“ (budget) b_i

– wird zu Beginn jeder Superperiode aufgefüllt auf „Kredit“ (allowence) a_i

– wird mit BM-Bedarf e_{ij} belastet, falls τ_{ij} zugelassen.

- **Systemstruktur**

  Basis-SRMS und Erweiterungen zur Leistungssteigerung:

  Zeitvererbung

  2. Chance

5.4. Basic-SRMS für harmonische Tasks

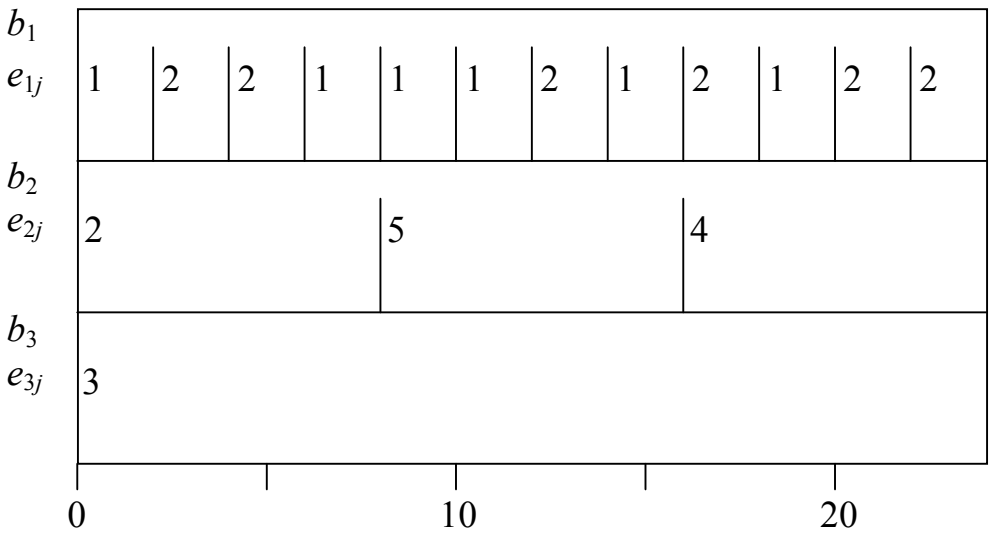
Harmonische Tasks (einfach-periodisch): $t_i < t_j \;\rightarrow\; t_i \mid t_j$

- **Lokale Zulassung von τ_{ij}**

$$\left(e_{ij} \le b_i\right) \wedge \left(e_{ij} \le t_i - \sum_{l=1}^{i-1} a_l \cdot \frac{t_i}{t_{l+1}}\right)$$

- **Beispiel.**

i	t_i	e_{ij}	$\bar{e}_{ij}$	u_i	$\bar{e}_{ij,\text{supper}}$	a_i	$\tilde{u}_i$
1	2	1..2					
2	8	1..7					
3	24	1..11					



5.5. Basic-SRMS für beliebige Tasks

- **Überlappender Job**

  Bereitzeit und Deadline liegen in verschiedenen Superperioden.

- **Lokale Zulassung**

  aufgrund Kontostands in Bereit-Superperiode:

  Job muß bis Ende dieser Periode beendet sein;

  aufgrund Kontostands in Deadline-Superperiode:

  Job muß bis Beginn dieser Periode verzögert werden.

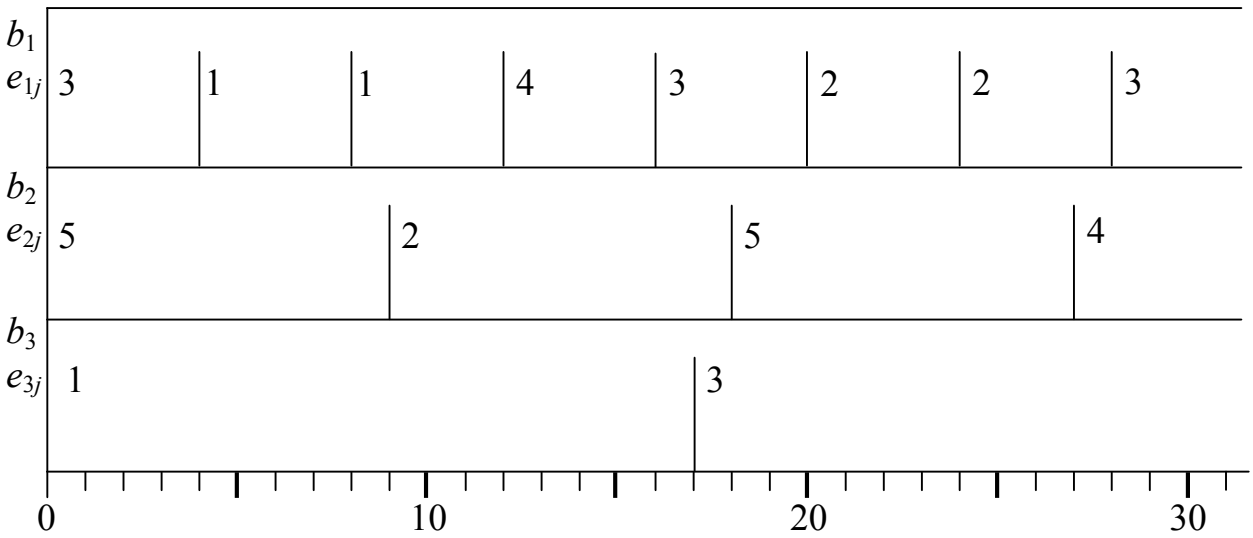
  In jedem Fall entsprechendes Konto belasten, ggf. auffüllen.

- **Zulassung von τ_{ij}**

$$\left(e_{ij} \le b_i\right) \wedge \left(e_{ij} \le t_i - \sum_{l=1}^{i-1}\left(a_l \cdot \left\lfloor \frac{t_i}{t_{l+1}} \right\rfloor + \min\left(a_l, t_i - t_{l+1} \cdot \left\lfloor \frac{t_i}{t_{l+1}} \right\rfloor\right)\right)\right)$$

- **Beispiel.**

t_i: 4; 9; 17 $\qquad$ a_i: 4; 8; -



5.6. Globale Zulassung für harmonische Tasks

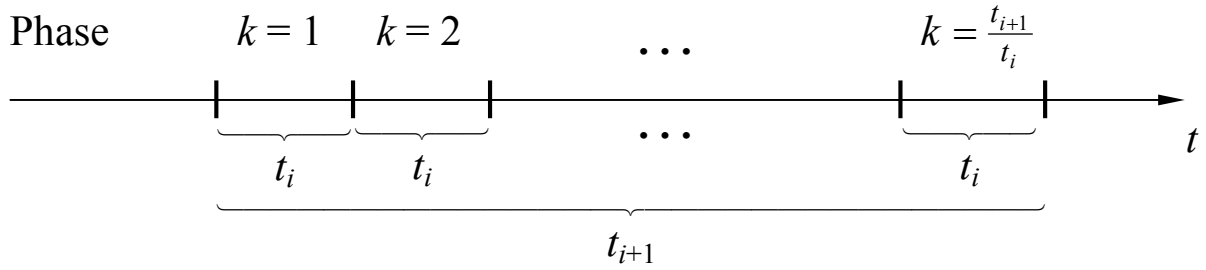
- ***T* einplanbar**

  Jedes $\tau_i \in T$ kann zu Beginn jeder Superperiode Kredit a_i erhalten.

  Kriterium: $$\sum_{i=1}^{n} \frac{a_i}{t_{i+1}} \leq 1$$

- **Berechnung von $\widetilde{Q}_i$:= QoS(a_i)**

– ***Taskphasen***



$$S_{ik} = \begin{cases} 1 & \tau_{ij} \text{ zu Beginn von Phase } k \text{ zugelassen} \\ 0 & \tau_{ij} \text{ zu Beginn von Phase } k \text{ abgewiesen} \end{cases}$$

– ***Berechnung von $\widetilde{Q}_i$***

$$\boldsymbol{P}(S_{i1} = 1) = \boldsymbol{P}(e_{ij} \leq a_i)$$

$$\boldsymbol{P}(S_{i2} = 1) = \boldsymbol{P}(e_{i,j\text{-}1} \leq a_i) \cdot \boldsymbol{P}(e_{i,j\text{-}1} + e_{ij} \leq a_i) + \boldsymbol{P}(e_{i,j\text{-}1} > a_i) \cdot \boldsymbol{P}(e_{ij} \leq a_i)$$

$\ldots$

$$\widetilde{Q}_i = \frac{t_i}{t_{i+1}} \cdot \sum_{k=1}^{t_{i+1}/t_i} P(S_{ik} = 1)$$

- **Berechnung von a_i aus $\widetilde{Q}_i$**

  $\widetilde{Q}_i$ = QoS(a_i) monoton wachsend $\rightarrow$

  Intervallschachtelung: kleinstes a_i, so daß $\widetilde{Q}_i \geq Q_i$

- **Beispiel.**

i	P_i	E_i^{max}	$E(E_i)$	PDF	# Phases
1	5	2	1.5	uniform	2
2	10	3	2.0	uniform	3
3	30	13	7.0	uniform	3
4	90	4	2.5	uniform	∞

Table 1: Example Task System

Guarantee Calculations for Task 1

a_1	$P(S_{1,1}=1)$	$P(S_{1,2}=1)$	$QoS(\tau_1)$
2	1.0000	0.2500	0.6250
4	1.0000	1.0000	1.0000

Guarantee Calculations for Task 2

a_2	$P(S_{2,1}=1)$	$P(S_{2,2}=1)$	$P(S_{2,3}=1)$	$QoS(\tau_2)$
3	1.0000	0.3333	0.2345	0.5230
6	1.0000	1.0000	0.6296	0.8770
9	1.0000	1.0000	1.0000	1.0000

Guarantee Calculations for Task 3

a_3	$P(S_{3,1}=1)$	$P(S_{3,2}=1)$	$P(S_{3,3}=1)$	$QoS(\tau_3)$
21	1.0000	0.9110	0.5628	0.8250
24	1.0000	0.9820	0.7010	0.8944
27	1.0000	1.0000	0.8340	0.9448
30	1.0000	1.0000	0.9250	0.9750
33	1.0000	1.0000	0.9745	0.9915
36	1.0000	1.0000	0.9950	0.9980
39	1.0000	1.0000	1.0000	1.0000

Table 2: QoS Calculations for the Example Task System shown in Table 1

a_1	a_2	a_3	a_4	Utilization	$QoS(\tau_1)$	$QoS(\tau_2)$	$QoS(\tau_3)$	$QoS(\tau_4)$
4	9	24	3	1.0000	1.0000	1.0000	0.8944	0.7500
4	3	39	4	0.9778	1.0000	0.5230	1.0000	1.0000
2	9	39	4	1.0000	0.6250	1.0000	1.0000	1.0000
4	6	33	3	1.0000	1.0000	0.8770	0.9915	0.7500

Table 3: Example allowance assignments with corresponding achievable QoS and utilization