Outline

Component-Based Software Design
Enrico Bini
Example
Linear supply
Simple application model
Richer application model

Component-Based Software Design

Hierarchical Real-Time Scheduling
lecture 4/4

Enrico Bini

March 25, 2015

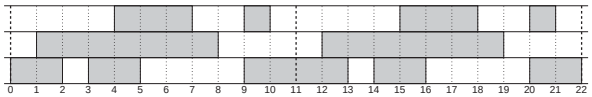
Component-Based Software Design
Enrico Bini
Example
Linear supply
Simple application model
Richer application model

Outline

Component-Based Software Design
Enrico Bini
Example
Linear supply
Simple application model
Richer application model

Example of computation

Given the following resource schedule with period 11



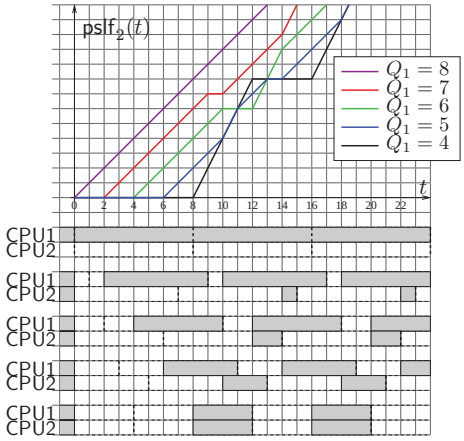
what are:

- what are $\mathsf{psf}_k(t)$?
- what is the max parallelism of the VM?

Component-Based Software Design
Enrico Bini
Example
Linear supply
Simple application model
Richer application model

Linear supply: example

$P = 8$, $Q_1 + Q_2 = 8$. What are $\beta_1, \beta_2, \Delta_1, \Delta_2$ of



Component-Based Software Design
Enrico Bini
Example
Linear supply
Simple application model
Richer application model

Linear supply bounds

- Supply lower bound functions $\mathsf{pslf}_k(t)$ can be lower bounded by a linear function (**add drawing**):

The supply lower bound function $\mathsf{pslf}_k(t)$ is lower bounded by any of the following functions

$$\mathsf{lpslf}_k(t) = \max\{0, \beta_k(t - \Delta_k)\}$$

with

$$\beta_k \leq \lim_{t\to\infty} \frac{\mathsf{pslf}_k(t)}{t}, \qquad \Delta_k \geq \sup_{t\geq 0} \left\{ t - \frac{\mathsf{pslf}_k(t)}{\beta_k} \right\}$$

- non-decr over k implies $\beta_{k+1} \geq \beta_k$
- concavity implies $\frac{\beta_k}{k} \geq \frac{\beta_{k+1}}{k+1}$
- must be $\Delta_{k+1} \leq \Delta_k$ to preserve non-decr on k

Outline

Linear supply: simplification

- Let's consider the simple case with $Q_1, \geq Q_2 \geq \ldots \geq Q_{\bar{m}}$, with period P
- The computation of the exact supply is challenging
- A valid lower bound is:

$$\mathsf{pslf}_k(t) \geq \sum_{\ell=1}^{k} \max\left\{0, \frac{Q_\ell}{P}(t - 2(P - Q_\ell))\right\}$$

since this bound is computed by considering the worst case for each budget independently

- Asymptotically the lower bound is

$$\frac{\sum_{\ell=1}^{k} Q_\ell}{P} t - 2\sum_{\ell=1}^{k} Q_\ell + 2\frac{\sum_{\ell=1}^{k} Q_\ell^2}{P}$$

If, for example, $\sum_{\ell=1}^{k} Q_\ell$ is constant, when is the linear lower bound maximized?

Serialization hypothesis

1. Parallel work can be serialized
   - Reasonable to assume
   - Notice in *gang scheduling* this is not the case: parallel work need to be scheduled **simultaneously** over several CPUs.
   - Gang scheduling is used to model parallel computation with tight interaction among threads

Application model

Definition
The work W is *malleable* if the time to complete over k physical machines is W/k, for all k.

Example: a set of many small jobs can be considered malleable (threads created by web servers to serve clients)
Several application models:

1. A *malleable* task with computation time C and deadline D (possibly $D < C$)
2. Set of n malleable tasks (C_i, T_i, D_i): T_i period, D_i deadline, C_i computation time (can be fully parallelized)
3. Set of n sequential tasks (C_i, T_i, D_i): T_i period, D_i deadline, C_i computation time ($\leq D_i$)
4. A *pipeline* with computation time C, period T ($< C$) and deadline D ($> C$)

One non-malleable task

Given a non-malleable (sequential) task with:

- computation time C and
- deadline D

Sequential work can exploit only one machine

- *worst-case response time* $R_{\mathsf{w}}(C)$ of sequential work C over a VM

$$R_{\mathsf{w}}(C) = \sup\{t : \mathsf{pslf}_1(t) < C\},$$

- *best-case response time* $R_{\mathsf{b}}(C)$ of sequential work C over a VM

$$R_{\mathsf{b}}(C) = \inf\{t : \mathsf{psuf}_1(t) \geq C\}.$$

- a sequential task schedulable if

$$\mathsf{pslf}_1(D) \geq C$$

One malleable task

Given a malleable task with:

- computation time C and
- deadline D (possibly $D < C$)

Malleable work can exploit any level of parallelism: no distiction between the two dimensions of resource

- *worst-case response time* $R_{\mathsf{w}}(C)$ of malleable work C over a VM

$$R_{\mathsf{w}}(C) = \sup\{t : \mathsf{pslf}_{\bar{m}}(t) < C\},$$

- *best-case response time* $R_{\mathsf{b}}(C)$ of malleable work C over a VM

$$R_{\mathsf{b}}(C) = \inf\{t : \mathsf{psuf}_{\bar{m}}(t) \geq C\}.$$

- malleable task schedulable if

$$\mathsf{pslf}_{\bar{m}}(D) \geq C$$

[Prove that if $D < \frac{C}{\bar{m}}$ then non-schedulable]

Set of malleable tasks

Theorem (EDF of malleable tasks)

A set task of n constrained deadline (with $D_i \le T_i$) malleable tasks is EDF-schedulable over a VM with $\mathsf{pslf}_{\bar m}(t)$, if

$$\forall t \in \mathcal{D} \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t+T_i-D_i}{T_i} \right\rfloor\right\} C_i \le \mathsf{pslf}_{\bar m}(t)$$

with

$$\mathcal{D} = \{d_{i,k} : d_{i,k} = kT_i + D_i,\ i = 1,\dots,n,\ k \in \mathbb{N},\ d_{i,k} \le D^*\}$$

and $D^* = \mathsf{lcm}(T_1,\dots,T_n) + \max_i\{D_i\}$.

Outline

Expression of the $W_i^{\mathcal{L}}$

- If local sched. algo. $\mathcal{L} = \mathsf{FP}$, then

$$W_i^{\mathsf{FP}} = \sum_{j \in \mathsf{hp}(i)} W_{ji},$$

where $\mathsf{hp}(i)$ denotes the set of indices of tasks with higher priority than i, and W_{ji} is the amount of interfering workload caused by j-th task on i-th task, that is

$$W_{ji} = N_{ji}C_j + \min\{C_j, D_i + D_j - C_j - N_{ji}T_j\}$$

with $N_{ji} = \left\lfloor \frac{D_i+D_j-C_j}{T_j} \right\rfloor$.

- If local sched. algo. $\mathcal{L} = \mathsf{EDF}$, then

$$W_i = \sum_{j=1, j\neq i}^{n} \left(\left\lfloor \frac{D_i}{T_j} \right\rfloor C_j + \min\left\{C_j, D_i - \left\lfloor \frac{D_i}{T_j} \right\rfloor T_j\right\}\right),$$

Set of non-malleable tasks

Theorem (Schedulability of non-malleable tasks)

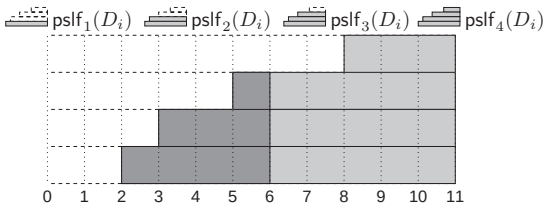
A set of n constrained deadline non-malleable tasks is schedulable by the local scheduling algorithm $\mathcal{L}$ over the VM abstracted by $\{\mathsf{pslf}_k\}_{k=1}^{\bar m}$, if

$$\bigwedge_{i=1,\dots,n} \bigvee_{k=1\dots,\bar m} k\,C_i + W_i^{\mathcal{L}} \le \mathsf{pslf}_k(D_i),$$

where $W_i^{\mathcal{L}}$ is the maximum interfering workload that can be experienced by i-th task in the interval $[0, D_i]$ with the local scheduling algorithm $\mathcal{L}$.

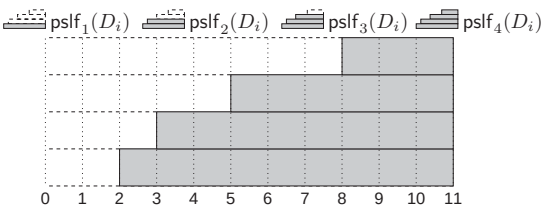
Proof sketch of Theorem 2/3

- The work W_i creates interference when it occupies all the available procesors
- Amount of interference created by ϵ work running at parallelism k is ϵ/k
- The created interference I_i is maximized when the processors are occupied by W_i starting from the lowest parallelism



If $W_i = 8$ then $I_i = 6$

Proof sketch of Theorem 1/3



Let us assume

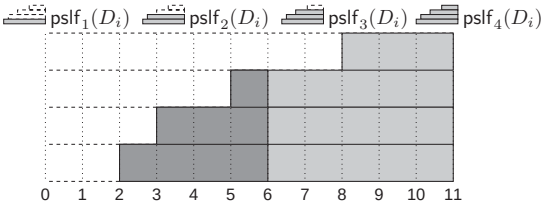
- $D_i = 11$, $C_i = 4$, $W_i = 8$
- $\bar m = 4$, $\mathsf{pslf}_1(11) = 9$, $\mathsf{pslf}_2(11) = 17$, $\mathsf{pslf}_3(11) = 23$, $\mathsf{pslf}_4(11) = 26$
- It the i-th task schedulable?
- How can W_i create as much interference as possible? **[Explain the intuition starting from W_i small]**

Comments on the Theorem

- Only sufficient condition. Sources of pessimism:
  1. in the accounting of the interfering workload W_i (the assumed scenario may never show up)
  2. the interfering workload is treated as *malleable* (it is assumed it can occupy any level of parallelism), while this is not the case.

Component-Based Software Design
Enrico Bini
Example
Linear supply
Simple application model
Richer application model

Proof sketch of Theorem 3/3



k^* be the max # of CPUs occupied by W_i ($k^* = 3$ above)

$$I_i = D_i - \frac{\mathsf{pslf}_{k^*}(D_i) - W_i}{k^*}.$$

By observing that the evaluation of the RHS for any other index $k \neq k^*$ is not smaller than I_i,

$$I_i = \min_{k=1,\ldots,m} \left\{ D_i - \frac{\mathsf{pslf}_k(D_i) - W_i}{k} \right\}.$$

[next steps on the whiteboard]

Component-Based Software Design
Enrico Bini
Example
Linear supply
Simple application model
Richer application model

Example of malleable task set

- Let us assume to have the following task set (C_i, T_i, D_i)
  - $\{(1,3,3),(1,4,4),(1,12,12)\}$
- Tasks are malleable. Harder or simpler than non-malleable?
- Local scheduler is EDF. Cond is

$$\forall t \in \mathcal{D} \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \right\} C_i \leq \mathsf{pslf}_{\bar{m}}(t)$$

- set of deadlines $\mathcal{D} = \{3,4,6,8,9,12\}$
- pairs are $(t,w) \in \{(3,1),(4,2),(6,3),(8,4),(9,5),(12,8)\}$
- How many CPUs of speed α?

Component-Based Software Design
Enrico Bini
Example
Linear supply
Simple application model
Richer application model

Example of malleable task set

- Let us assume to have the following task set (C_i, T_i, D_i)
  - $\{(1,3,3),(1,4,4),(1,12,12)\}$
- Tasks are malleable. Harder or simpler than non-malleable?

Component-Based Software Design
Enrico Bini
Example
Linear supply
Simple application model
Richer application model

Example of non-malleable task set

- Let us assume to have the following task set (C_i, T_i, D_i)
  - $\{(1,3,3),(1,4,4),(1,12,12)\}$
- Tasks are malleable (can exploit any parallelism)
- Local scheduler is EDF. Cond is

$$\bigwedge_{i=1,\ldots,n} \bigvee_{k=1\ldots,\bar{m}} k\,C_i + W_i^{\mathsf{EDF}} \leq \mathsf{pslf}_k(D_i),$$

with

$$W_i^{\mathsf{EDF}} = \sum_{j=1, j\neq i}^{n} \left(\left\lfloor \frac{D_i}{T_j} \right\rfloor C_j + \min\left\{ C_j, D_i - \left\lfloor \frac{D_i}{T_j} \right\rfloor T_j \right\} \right),$$

- **[Illustration of the condition over the $\mathsf{pslf}_k(t)$ plane]**

Component-Based Software Design
Enrico Bini
Example
Linear supply
Simple application model
Richer application model

Example of malleable task set

- Let us assume to have the following task set (C_i, T_i, D_i)
  - $\{(1,3,3),(1,4,4),(1,12,12)\}$
- Tasks are malleable. Harder or simpler than non-malleable?
- Local scheduler is EDF. Cond is

$$\forall t \in \mathcal{D} \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \right\} C_i \leq \mathsf{pslf}_{\bar{m}}(t)$$

- set of deadlines $\mathcal{D} = \{3,4,6,8,9,12\}$
- pairs are $(t,w) \in \{(3,1),(4,2),(6,3),(8,4),(9,5),(12,8)\}$
- How many CPUs of speed α?

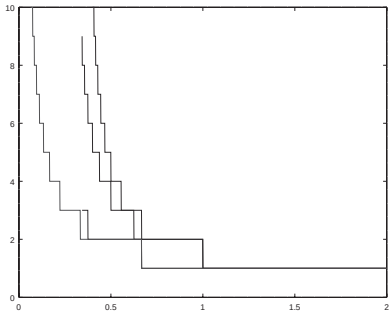
$\mathsf{pslf}_{\bar{m}}(t) = \alpha \bar{m} t$

point $(12,8)$ is dominating

$$\alpha \bar{m} 12 \geq 8 \qquad \Rightarrow \qquad \bar{m} \geq \left\lceil \frac{2}{3\alpha} \right\rceil$$

Comparing the malleable vs. non-malleable

- number of minimum CPUs as function of the speed



Example of non-malleable task set

- How many CPUs of speed α?

C_i	T_i	D_i	W_i^{EDF}
1	3	3	2
1	4	4	$1+2=3$
1	12	12	$7+0=7$

- Condition is

$$\bigwedge_{i=1,\ldots,n} \bigvee_{k=1\ldots,\bar{m}} k\,C_i + W_i^{\text{EDF}} \le \text{pslf}_k(D_i) = k\alpha D_i$$

- First, it must necessarily be

$$\forall i,\ \alpha > \frac{C_i}{D_i} \quad \Rightarrow \quad \alpha > \max_i \left\{ \frac{C_i}{D_i} \right\}$$

otherwise, impossible to schedule single tasks with $W_i = 0$.

- Then

$$k \ge \max_i \left\lceil \frac{W_i}{\alpha D_i - C_i} \right\rceil = \bar{m}$$

Feasible VM speeds: example

$$\bigwedge_{i=1,\ldots,n} \bigvee_{k=1\ldots,\bar{m}} \sum_{\ell=1}^{k} \alpha_\ell \ge \frac{k\,C_i + W_i^{\mathcal{L}}}{D_i}$$

C_i	T_i	D_i	W_i
1	3	3	2
1	4	4	$1+2=3$
1	12	12	$7+0=7$

Feasible VM speeds

- Let us assume that the VM receives a set of $\bar{m}$ budgets $Q_1, Q_2, \ldots, Q_{\bar{m}}$ every period P. We assume $Q_1 \ge Q_2 \ge \ldots \ge Q_{\bar{m}}$
- To simplify the analysis, we assume that $P, Q_k \to 0$, with constant

$$\alpha_k = \frac{Q_k}{P}$$

- This implies

$$\text{pslf}_k(t) = \left(\sum_{\ell=1}^{k} \alpha_\ell \right) t$$

- Then the feasible speeds of the CPUs are

$$\bigwedge_{i=1,\ldots,n} \bigvee_{k=1\ldots,\bar{m}} \sum_{\ell=1}^{k} \alpha_\ell \ge \frac{k\,C_i + W_i^{\mathcal{L}}}{D_i}$$

Feasible Q_i: example

By exploiting the lower bound on the $\text{pslf}_k(t)$ of

$$\text{pslf}_k(t) \ge \sum_{\ell=1}^{k} \max\left\{0, \frac{Q_\ell}{P}(t - 2(P - Q_\ell))\right\}$$

we can find a sufficient on the Q_i that guarantee a given task set

$$\bigwedge_{i=1,\ldots,n} \bigvee_{k=1\ldots,\bar{m}} k\,C_i + W_i^{\mathcal{L}} \le \text{psfl}_k(D_i)$$

C_i	T_i	D_i	W_i
1	3	3	2
1	4	4	$1+2=3$
1	12	12	$7+0=7$