Real-Time Systems (for PhD students)

Enrico Bini

2024, Unito

Contents

1 Introduction to the course

Presentation

- Course for PhD: evolving, open problems, there may be mistakes, it requires interaction
- Goal: to give a broad view of the real-time systems area at large
- Lecture style: expose quickly many topics ⇒ you better slow me down by asking question!
- Schedule of classes (subscribe to the Google Calendar)[2pt]

Tue	Apr 09	15:00–17:00	Sala Sem. Primo piano
Fri	Apr 12	14:00–16:30	Sala Sem. Primo piano
Tue	Apr 23	14:30–17:00	Sala Riun. Primo pian
Fri	Apr 24	14:30–17:00	Sala Riun. Primo pian
...	...	...	...

- Evaluation:
  - a homework: about one week of time will be given
  - the presentation of an assigned research paper
- On Tuesday, April 23, each student **presents his/her research area**
  - for brainstorming about the to-be-chosen paper to study

Behaviour of computing systems

- Logic gates (AND, OR, NOT...) have a deterministic behavior
  - (actually, new integration technologies are challenging even the behavior of basic logic gates)
- The exact model of a computing system by analyzing at the level of logic gates is infeasible
- Exactness is forgotten and **uncertainties** are introduces
- Sources of uncertainties
  1. input: what data will a sensor read?
  2. state: what will be the state (registers, cache, memory) when executing some code?

Model of uncertainties

Different approaches to uncertainties

- Benchmarking: "we try a couple of references applications to imagine the behaviour in general"
  - pros: easy to implement, the only feasible on general purpose computing
  - cons: what can we really say in general?
- Stochastic models: "we attach a probability measure onto the uncertainties"
  - pros: analytic results are possible via (stochastic) formal methods, queueing theory, etc.
  - cons: what is the probability measure of the input? What about the cache content?
- Bounding uncertainties with intervals, polytopes, etc.
  - pros: analytic results are possible via (deterministic) formal methods, real-time scheduling, etc.
  - cons: results may be very conservative as they may be dominated by corner cases which do not appear in reality

2 Scheduling Problems

Basic ingredients of scheduling

- a *time set* $\mathcal{T}$ (typically $\mathbb{N}$ or $[0, \infty)$);
- a set $\mathcal{N}$ of *tasks* (aka demands, works, jobs) requiring work to be made. Since they are finite, we represent them by $\mathcal{N} = \{1, 2, \ldots, n\}$;
- a set $\mathcal{M}$ of *machines* (aka processors, resources, workers, etc.) capable to perform some work (one/many machine, heterogeneous multicore, etc.). Since they are finite we represent them by $\mathcal{M} = \{1, 2, \ldots, m\}$;
- a set $\mathcal{C}$ of *constraints*, such as:
  1. affinities: "task i may execute only over $\mathcal{M}_i \subseteq \mathcal{M}$
  2. deadlines: "task i must complete not later than a deadline D"
  3. mutual exclusion: "task i may not execute while task j is executing"
  4. parallelism of tasks: "task i needs 3 processors in parallel when executing
- a *scheduling algorithm* $\mathcal{A}$ which produces a *schedule* S of $\mathcal{N}$ over $\mathcal{M}$, compatible with $\mathcal{C}$.

Task 1/2

Characteristics of tasks:

- amount of work,
- recurrent/non-recurrent, does a task repeat over time? How often?
- on-line/off-line, do we know the parameters in advance?
- sequential (only one machine at time), parallel (more than one machine at time), parallelizable (one or more machines at time).

Task 2/2

- we denote the *work request* of the i task by $r_i(t)$. If the work is composed by a sequence of jobs of sizes $\{C_{i,j}\}$, requested at instants $\{a_{i,j}\}$, then
$$r_i(t) = \sum_j \delta(t - a_{i,j}) C_{i,j}$$
with $\delta(t - t_0)$ is a unitary Dirac's delta at t_0.
  - Notice that we may use $C_{i,j} < 0$ to indicate the suppresion of some work. (although never in this course)
- we denote the *pending work* of the i task by $w_i(t)$

Basic principle[1]

$$\forall t_0, t_1 \geq t_0, \quad w_i(t_1) \leq w_i(t_0) + \int_{t_0}^{t_1} r_i(t)\, dt$$

Machine

Characteristics of a machine:

- type, (CPU, GPU, etc.)
- speed $\sigma_k(t)$ of machine k at time t, which may be time-varying;
- operating modes (variable speed over time, etc.)
- availability of dedicated resources (printer, some dedicated hardware, etc.)

[1] to avoid ambiguity, $\int_a^b f(t)\, dt = \int_{[a,b)} f(t)\, dt$

Constraints

- precedence constraints: "this task can start only after this other task"
- deadlines: "the work must be completed by this instant",
- affinity to machines "this task must always run over this subset of machines"
- mutual exclusion, "these portions of tasks cannot be scheduled at the same time"

Schedule

A schedule is a function

$$S : \mathcal{M} \times \mathcal{T} \to \mathcal{N} \cup \{0\}$$

with "task 0" denoting *idle*, compatible with the constraints in $\mathcal{C}$.

If $m = |\mathcal{M}| = 1$ (one machine) then just $S : \mathcal{T} \to \mathcal{N} \cup \{0\}$.

- If $S(k, t) = i$ then the machine k is assigned to the i-task at time t.
- If $S(k, t) = 0$ then the machine k is not assigned at time t (we say that the k-th machine is *idle* at t).
- This definition of schedule implies that at every instant t each machine is assigned to at most one task.
- Conversely, at every instant each task may be assigned any number of machines in $\mathcal{M}$. (possible in case of task parallelism)

Example of schedule

- The inverse image of i under S, $S^{-1}(i) \subseteq \mathcal{M} \times \mathcal{T}$, that is

  $$S^{-1}(i) = \{(k, t) \in \mathcal{M} \times \mathcal{T} : S(k, t) = i\}$$

  represents the machines allocated to the i-th task.
- Draw a task schedule
- the i-th task is *sequential* if

  $$\forall (k, t), (k', t') \in S^{-1}(i),\ t = t', \quad \Rightarrow \quad k = k'$$

  then we can define $s_i(t)$ indicator function of
- a schedule S is *partitioned* if

  $$\forall i, \forall (k, t), (k', t') \in S^{-1}(i), \quad k = k'$$

- task i has *affinity* $\mathcal{M}_i \subseteq \mathcal{M}$ if

  $$\forall (k, t) \in S^{-1}(i), \quad k \in \mathcal{M}_i$$

Conservation of work

Law of "conservation of work"

$$\forall t_0 < t_1, \quad w_i(t_1) = w_i(t_0) - s_i(t_0, t_1) + \int_{t_0}^{t_1} r_i(t)\, dt$$

with the *scheduled resource* $s_i(t_0, t_1)$ to the i task, defined by

$$s_i(t_0, t_1) = \int_{t_0}^{t_1} \sum_{(k,t) \in S^{-1}(i)} \sigma_k(t)\, dt$$

with $\sigma_k(t)$, speed of k-th machine, at time t. If machines are identical then $\sigma_k = 1$

Scheduling algorithms

Goal of scheduling algorithm $\mathcal{A}$: find a schedule S such that:

- the constraints are met (in this case constraints have to be specified). Example: all task deadlines are met,
- some target function is minimized/maximized (minimum makespan/delay, best "performance": requires to know how the timing affect the "performance")

Characteristics of scheduling algorithms:

- *work-conserving*: no idle machine if pending tasks exist;

$$\forall k \in \mathcal{M},\ S(k,t) = 0 \quad \Rightarrow \quad \forall i \in \mathcal{N}, w_i(t) = 0$$

- *preemptive*, $\mathcal{A}$ can interrupt a task while it executes;
- time-complexity: how long does it take to decide the machine assignment?
- clairvoyant: takes decisions based on the knowledge of the future

Examples of scheduling algorithms

Examples of scheduling algorithms:

- First In First Out (FIFO), schedule tasks in order of arrivals;
- Round Robin (RR), divide the time in slices and assign slices in round;
- Shortest Job First (SJF) and its preemptive version Shortest Remaining Time First (SRTF);
- Earliest Deadline First (EDF), assigns priority according to the deadlines d;
- Least Laxity First (LLF), aka Least Slack Time (LST), at t assigns priority according to the smallest "laxity" $(d - t) - w(t)$
- Fixed Priorities (FP), tasks are prioritized.

Feasibility vs. Schedulability

Definition 1. A task set $\mathcal{N}$ is *feasible* is it exists a schedule which satisfies the task constraint.

Definition 2. A task set $\mathcal{N}$ is *schedulable* by the scheduling algorithm $\mathcal{A}$, if $\mathcal{A}$ can produce a schedule S which does not violate any constraint of $\mathcal{C}$.

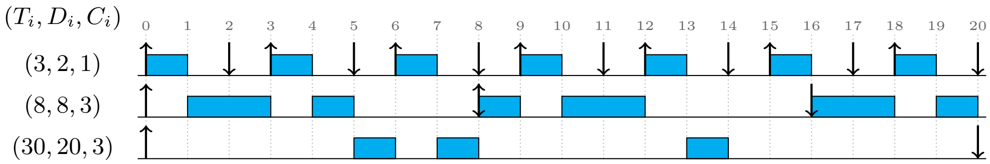
- Goal of scheduling algorithm design: to design a scheduling algorithm $\mathcal{A}$ which can schedule any feasible task set.

Analysis, sensitivity, and design

- Given a scheduling algorithm $\mathcal{A}$, we distinguish three problems (of increasing difficulty):
  1. **analysis**: given a set $\mathcal{N}$ of tasks and a set $\mathcal{M}$ of machines, is $\mathcal{N}$ schedulable by $\mathcal{A}$ over $\mathcal{M}$?
  2. **sensitivity**: given a set $\mathcal{N}$ of tasks schedulable by $\mathcal{A}$ over $\mathcal{M}$. How much can we modify $\mathcal{N}$ such that it remains schedulable?
  3. **optimal design**: given a set $\mathcal{N}$ of tasks. What is the best set of machines $\mathcal{M}$ such that $\mathcal{N}$ is schedulable by $\mathcal{A}$ over $\mathcal{M}$?

3 Task Model

Real-Time (RT) Task Model

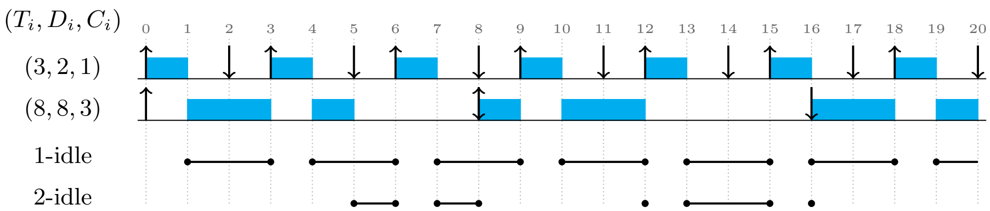


- *Task i* denoted by τ_i
- Tasks are indexed by decreasing *priority*: τ_1 highest, τ_n lowest
- Tasks recurrently release *jobs*
- Jobs with higher priority may *preempt* other
- The releases of consecutive jobs are separated by at least T_i (called *period*)
- All jobs belonging to τ_i have an *execution time* C_i;
- The task *utilization* is $U_i = C_i/T_i$ denotes the fraction of time needed by τ_i
- All jobs belonging to τ_i have a relative *deadline* $D_i \le T_i$ (constrained deadline)

Level-i busy, idle, etc.

Definition 3 (Request bound function)**.** The *level-i request bound function* $\mathsf{rbf}_i(t) = \sum_{j=1}^{i} \left\lceil \frac{t}{T_j} \right\rceil C_j$ is the maximum amount of work which can be **requested** by tasks with priority i or higher in any interval of length t.

Definition 4. The *level-i idle time* is the set of instants when no tasks in $\{1, \dots, i\}$ is running or ready. (execution intervals are open: not including extremes. idle intervals closed)



4 FP exact analysis: response time

Computing R_i: interference

Definition 5. We define the *level-i interference* $I_i(t)$ as the maximum amount of work which can be requested by tasks with priority higher than i in any interval of length t.

For our simple task model,

$$I_i(t) = \sum_{j=1}^{i-1} \int_0^t r_j(t)\, dt = \sum_{j=1}^{i-1} \int_0^t C_j \sum_{\ell} \delta(t - \ell T_j)\, dt$$
$$= \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j$$

and it corresponds to the scenario with all tasks activated together at 0 at the highest possible rate.

- Example of interference with different activation patterns (two alternating periods)

Computing R_i: recurrent equation

- The response time of the first job of τ_i is found as the smallest fixed point of the following equation [11]

$$\begin{cases} R_{i,1}^{(0)} = C_i \\ R_{i,1}^{(k+1)} = C_i + I_i(R_{i,1}^{(k)}) \end{cases} \tag{1}$$

- If $\sum_{j=1}^{i-1} U_j < 1$, then it converges
- Explain its rationale.

5 FP: Sensitivity Analysis

Scheduling points tests

An alternate exact test enables sensitivity analysis.

Theorem 6 (Lehoczky et al. [13]). *A constrained deadline (with $D_i \le T_i$) task set is schedulable by FP if and only if*

$$\forall i \in \mathcal{N},\ \exists t \in [0, D_i], \quad C_i + I_i(t) \le t,$$

with $I_i(t)$ being the level-i interference.[2]

It tasks are sporadic then

$$I_i(t) = \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j$$

Interesting, but not so practical (how do we check if it exists a point in a real interval?)

First simple reduction

It tasks are sporadic (with period T_i), then the previous condition is equivalent to the following one, which can be better managed

$$\forall i \in \mathcal{N},\ \exists t \in \mathcal{S}_i, \quad C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \le t$$

with

$$\mathcal{S}_i = \{kT_j : k \in \mathbb{N},\ 0 < kT_j < D_i,\ j < i\} \cup \{D_i\}$$

This is the set of *scheduling points*.

However the points in $\mathcal{S}_i$ can still be many and period dependent, especially when the periods of the higher priority tasks are significantly smaller than D_i.

Second sophisticated reduction

- Can we remove points from $\mathcal{S}_i$ to reduce the complexity?
- By removing arbitrarily points we may lose necessity.
- However it is possible [6] to remove many points without losing necessity.

Theorem 7. *A constrained deadline (with $D_i \le T_i$) task set is schedulable by FP if and only if [6]*

$$\forall i \in \mathcal{N},\ \exists t \in \mathcal{P}_{i-1}(D_i), \quad C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \le t \tag{2}$$

with $\mathcal{P}_j(t)$ being a set recursively defined as

$$\begin{cases} \mathcal{P}_0(t) = \{t\} \\ \mathcal{P}_j(t) = \mathcal{P}_{j-1}\left(\left\lfloor \frac{t}{T_j} \right\rfloor T_j\right) \cup \mathcal{P}_{j-1}(t). \end{cases} \tag{3}$$

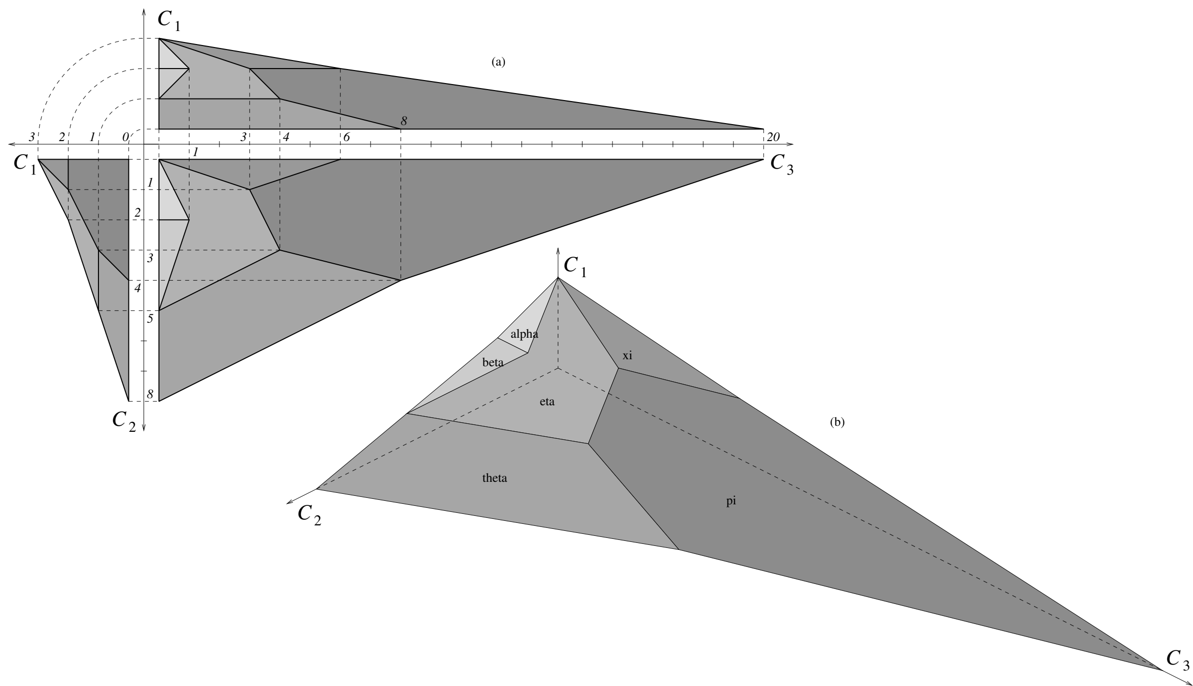
Show how $\mathcal{P}_{i-1}(D_i)$ is computed ($D_1 = 3, T_1 = 4, D_2 = 10$) C_i-plane.

[2] the maximum amount of work which can be requested by tasks with priority higher than i in any interval of length t

Visualization of the test

$$\forall i \in \mathcal{N},\ \exists t \in \mathcal{P}_{i-1}(D_i), \quad C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \le t \tag{4}$$

If $T_1 = 3, T_2 = 8, T_3 = 20$, and $D_i = T_i$ the schedulable C_i are



Sensitivity: min schedulable speed

- If the processor runs at speed r, then all computation times become C_i/r
- From the scheduling point condition it is not difficult to find [8] the smallest speed that guarantee FP-schedulability

$$\forall i \in \mathcal{N},\ \exists t \in \mathcal{P}_{i-1}(D_i), \quad \frac{C_i}{r} + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil \frac{C_j}{r} \le t$$

$$\forall i \in \mathcal{N},\ \exists t \in \mathcal{P}_{i-1}(D_i), \quad r \ge \frac{C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j}{t}$$

$$r \ge \max_{i \in \mathcal{N}} \min_{t \in \mathcal{P}_{i-1}(D_i)} \frac{C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j}{t}$$

- (there is no direct way to find it using the response time)
- Example of calculation for two tasks in the plot.

Sensitivity: max schedulable C_k

What is the maximum schedulable C_k?

- all tasks τ_i with $i < k$ are unaffected by C_k
- to ensure the schedulability of τ_k it must be

$$C_k \le \max_{t \in \mathcal{P}_{k-1}(D_k)} (t - \sum_{j=1}^{k-1} \left\lceil \frac{t}{T_j} \right\rceil C_j)$$

The RHS is the amount of *level-*$(k-1)$ *idle time* in $[0, D_k]$

- to ensure the schedulability of all tasks with lower priority $i > k$ it must be

$$C_k \leq \min_{i=k+1,\ldots,n} \max_{t \in \mathcal{P}_{i-1}(D_i)} \frac{t - (C_i + \sum_{\substack{j=1 \\ j \neq k}}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j)}{\left\lceil \frac{t}{T_k} \right\rceil}$$

- hence $C_k^{\max}$ is the minimum of the two RHS

6 Earliest Deadline First: basics

Earliest Deadline First

- Task model is still the same $\tau_i = (C_i, T_i, D_i)$;
- In FP, priorities are per task: all jobs of same task that have the same priority;
- In EDF, priorities are per job: jobs are prioritized according to their absolute deadline.

Draw the EDF schedule of $[(2, 4), (3, 6)]$.

Most interesting feature

Theorem 8 (Liu and layland, 1973 [16]). *If a task set is feasible, then it is EDF-schedulable.*

Theorem 9 (Liu and Layland, 1973 [16]). *If $D_i = T_i$ (implicit deadline) then a task set is EDF-schedulable **if and only if**:*

$$\sum_{i=1}^{n} U_i \leq 1$$

- Any FP-schedulable task set is also EDF-schedulable task set.

7 EDF: demand bound function

Demand bound function

If $D_i \neq T_i$ the schedulability condition becomes more complicated.

Theorem 10 (Lemma 3 in [4]). *The task set $\mathcal{N}$ is EDF-schedulable **if and only if**:*

$$\forall t \geq 0 \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i \leq t$$

The LHS is called *demand bound function* $\mathsf{dbf}(t)$ of the task set at t.

- $\mathsf{dbf}(t)$ is the maximum amount of work of jobs with both activation and deadline in $[0, t]$.
- no per-task condition: any task may influence others
- $\max\{0, \cdot\}$ only needed for i with $D_i > T_i$

Making it more practical

- Obviously, checking $\forall t > 0$ is not very practical
- By observing the step shape of the `dbf` we can check only at the points where the steps occur

Theorem 11 (Lemma 3 in [4])**.** *The task set $\mathcal{N}$ is EDF-schedulable* ***if and only if*** $\sum_i U_i \leq 1$ *and:*

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

with

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

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

$H = \mathsf{lcm}(T_1, \ldots, T_n)$ is often called *hyperperiod* of the task set.

Mention the work by Ripoll [20] to compute the busy period, and Spuri [23] to compute the response time unde EDF.

Reducing the number of points

- If $U < 1$, then the for large t the condition is always true
- then D^* can be computed [4] by upper bounding $\mathsf{dbt}(t)$ with a line and we find

$$D^* = \frac{U}{1-U} \max_i\{T_i - D_i\}$$

- what happen to D^* if $\max_i\{T_i - D_i\} \leq 0$?
- the task set is obviously EDF-schedulable,
  - $\max_i\{T_i - D_i\} \leq 0 \quad \Leftrightarrow \quad \forall i,\ D_i \geq T_i$
  - EDF is sustainable, hence if $D_i = T_i$ is sched then $D_i \geq T_i$ also sched
  - Since $U < 1$, the task set is sched

Faster exact test

- All deadlines in $[0, D^*]$ may be too many
- Zhang and Burns [24] proposed the Quick convergence Processor-demand Analysis (QPA)

$d_{\min} \leftarrow \min\{D_i\}$
$t \leftarrow \max\{d_{i,k} : d_{i,k} \leq D^*\}$ ▷ initial assignment
while $\mathsf{dbf}(t) \leq t \wedge \mathsf{dbf}(t) > d_{\min}$ **do**
    if $\mathsf{dbf}(t) < t$ **then**
        $t \leftarrow \mathsf{dbf}(t)$
    else
        $t \leftarrow \max\{d_{i,k} : d_{i,k} < t\}$ ▷ escape from fixed points
    end if
end while
if $\mathsf{dbf}(t) \leq d_{\min}$ **then** task set EDF-schedulable
else task set not EDF-schedulable
end if

8 EDF: sufficient tests

Sufficient tests

- By replacing T_i with the more conservative value $\min\{T_i, D_i\}$, we find

$$\sum_{i=1}^{n} \frac{C_i}{\min\{T_i, D_i\}} \leq 1$$

  the ratio $\frac{C_i}{\min\{T_i,D_i\}}$ is often called *density* of the task

More sophisticated suff test

Devi proposed the following sufficient test

- Assuming that tasks are sorted by non-decreasing relative deadline ($D_i \leq D_2 \leq \ldots \leq D_n$)

Theorem 12 (Theorem 1 in [9]). *The task set $\mathcal{N}$ is EDF-schedulable if:*

$$\forall k = 1, \ldots, n \quad D_k \sum_{i=1}^{k} U_i + \sum_{i=1}^{k} \frac{T_i - \min\{T_i, D_i\}}{T_i} C_i \leq D_k$$

- Proved to strictly dominate the density test
- Linear complexity

FPTAS for EDF

- Albers et al. [1] proposed a Fully Polynomial Time Approximation Scheme.
- The i-th term in $\mathsf{dbf}(t)$ can be upper bounded by

$$\max\{0, \left\lfloor \frac{t - D_i + T_i}{T_i} \right\rfloor\} C_i \leq \mathsf{dub}_i(k,t) = \begin{cases} \max\{0, \left\lfloor \frac{t-D_i+T_i}{T_i} \right\rfloor\} C_i & t \leq d_{i,k} = (k-1)T_i + D_i \\ U_i(t + T_i - D_i) & t > d_{i,k} \end{cases}$$

so that

$$\frac{k}{k+1} \sum_i \mathsf{dub}_i(k,t) \leq \mathsf{dbf}(t) \leq \sum_i \mathsf{dub}_i(k,t)$$

FTPAS for EDF

- This enables a quite interesting result

  Theorem 13. *If $U \leq 1$ and*

  $$\forall t \in \mathcal{D}(\overline{k}) = \{d_{i,k} \in \mathbb{R}^+ : d_{i,k} = (k-1)T_i + D_i, 1 \leq k \leq \overline{k}\} \sum_{i=1}^{n} \mathsf{dub}_i(\overline{k}, t) \leq t$$

  then the task set is schedulable.

  Otherwise it is not schedulable over a CPU with speed $\frac{\overline{k}}{\overline{k}+1}$.

- In this way only $n\overline{k}$ evaluation of the dbf are needed.
- We can trade accuracy vs. complexity. As $\overline{k} \to \infty$ it becomes necessary and sufficient.
- FPTAS

Similar result can be derived for FP.

9 EDF: sensitivity analysis

Min EDF-schedulable speed

Similarly as in the FP case we can find the minimum EDF-schedulable speed as follows

$$r^{\mathsf{min}} = \max_{t \in \mathcal{D}} \frac{\sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t+T_i-D_i}{T_i} \right\rfloor\right\} C_i}{t}$$

However $D^* = H + \max_i\{D_i\}$, because we are changing the speed and then altering the linear upper bound that motivates the expression

$$D^* = \frac{U}{1-U} \max_i\{T_i - D_i\}$$

Max EDF-schedulable comp time

The maximum EDF-schedulable $C_k^{\max}$ can be computed in a similar way as in FP

$$C_k^{\max} = \min_{t \in \mathcal{D}, t \geq D_k} \frac{t - \sum_{\substack{i=1 \\ i \neq k}}^n \max\left\{0, \left\lfloor \frac{t+T_i-D_i}{T_i} \right\rfloor\right\} C_i}{\left\lfloor \frac{t+T_k-D_k}{T_k} \right\rfloor}$$

Here too, $D^* = H + \max_i\{D_i\}$.

10 Example

Sufficient tests

Let us have the following task set

T_i	D_i	C_i
3	5	1
8	8	2
20	10	5

Not DM-schedulable since $R_3 = 14 > D_3 = 10$

- Density test

$$\frac{1}{3} + \frac{2}{8} + \frac{5}{10} = \frac{13}{12} > 1$$

- Devi's test

$$\begin{aligned}
k = 1 \quad & D_1 U_1 \leq D_1 \qquad \Rightarrow \quad \text{OK} \\
k = 2 \quad & D_2(U_1 + U_2) \leq D_2 \quad \Rightarrow \quad \text{OK} \\
k = 3 \quad & D_3(U_1 + U_2 + U_3) + (T_3 - D_3)U_3 \leq D_3 \\
& U_1 + U_2 + \frac{C_3}{D_3} \leq 1 \quad \Rightarrow \quad \text{NO}
\end{aligned}$$

Example

Exact test:

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

with

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

Computing the upper bound D^* to the set of deadlines $\mathcal{D}$

$$D^* = \frac{U}{1-U} \max T_i - D_i = \frac{\frac{5}{6}}{\frac{1}{6}} 10 = 50$$

$$\begin{aligned}
\mathcal{D} = \{&5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, \\
&8, 16, 24, 32, 40, 48, 10, 30, 50\}
\end{aligned}$$

Example: QPA

$$\begin{aligned}
& d_{\min} = 5 \\
t = 50, \quad & \mathsf{dbf}(50) = 16 + 6 \times 2 + 3 \times 5 = 43 \\
t = 43, \quad & \mathsf{dbf}(43) = 13 + 5 \times 2 + 2 \times 5 = 33 \\
t = 33, \quad & \mathsf{dbf}(33) = 10 + 4 \times 2 + 2 \times 5 = 28 \\
t = 28, \quad & \mathsf{dbf}(28) = 8 + 3 \times 2 + 5 = 19 \\
t = 19, \quad & \mathsf{dbf}(19) = 5 + 2 \times 2 + 5 = 14 \\
t = 14, \quad & \mathsf{dbf}(14) = 4 + 2 + 5 = 11 \\
t = 11, \quad & \mathsf{dbf}(11) = 3 + 2 + 5 = 10 \\
t = 10, \quad & \mathsf{dbf}(10) = 2 + 2 + 5 = 9 \\
t = 9, \quad & \mathsf{dbf}(9) = 2 + 2 = 4 \\
& \text{exit because } \mathsf{dbf}(9) = 4 < d_{\min} = 5
\end{aligned}$$

dbf computed 9 times, instead of $|\mathcal{D}| = 22$ times

Example: FPTAS

$$\begin{aligned}
\overline{k} = 1 \quad & \mathcal{D}(1) = \{5, 8, 10\} \\
t = 5 \quad & 1 + 0 + 0 \leq 5 \\
t = 8 \quad & 2 + 2 + 0 \leq 8 \\
t = 10 \quad & 2.666 + 2.5 + 5 > 10 \\
\overline{k} = 2 \quad & \mathcal{D}(2) = \{5, 8, 10, 16, 30\} \\
t = 5 \quad & 1 + 0 + 0 \leq 5 \\
t = 8 \quad & 2 + 2 + 0 \leq 8 \\
t = 10 \quad & 2.666 + 2 + 5 \leq 10 \\
t = 16 \quad & 4.666 + 4 + 5 \leq 16 \\
t = 30 \quad & 9.333 + 7.5 + 10 \leq 30
\end{aligned}$$

- EDF-schedulable
- non EDF-schedulable over a speed $\frac{1}{2}$ processor

11 EDF: space of comp times

Space of feasible C_i

Theorem 14 (Lemma 3 in [4])**.** *$\mathcal{N}$ is EDF-schedulable* ***if and only if*** *$\sum_i U_i \leq 1$ and:*

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

with

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

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

$H = \mathsf{lcm}(T_1, \ldots, T_n)$ is often called the *hyperperiod.*

- Since we are investigating the space of C_i, no smaller D^* can be considered.
- For given T_i and D_i, the space of EDF-schedulable C_i is convex!

An example

Let us assume $T_1 = 4, D_1 = 5$ and $T_2 = 6, D_2 = 5$.

$$\mathcal{D} = \{5, 9, 11, 13, 17\}$$

Hence equations are

$$\begin{bmatrix} 4 & 6 \\ 8 & 6 \\ 8 & 12 \\ 12 & 12 \\ 16 & 18 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \leq \begin{bmatrix} 5 \\ 9 \\ 11 \\ 13 \\ 17 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

An example: viewing the C_i

Reducing the set of deadlines

Hence only the equations

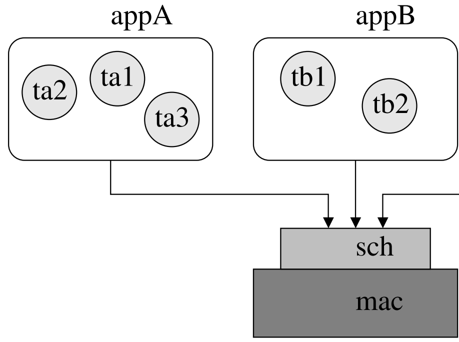
$$\begin{bmatrix} 4 & 6 \\ 16 & 18 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \leq \begin{bmatrix} 5 \\ 17 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

are needed to be checked, corresponding to $\mathcal{D} = \{5, 17\}$

FIXME: add here cutting deadline

12 Hierarchical Scheduling: intro

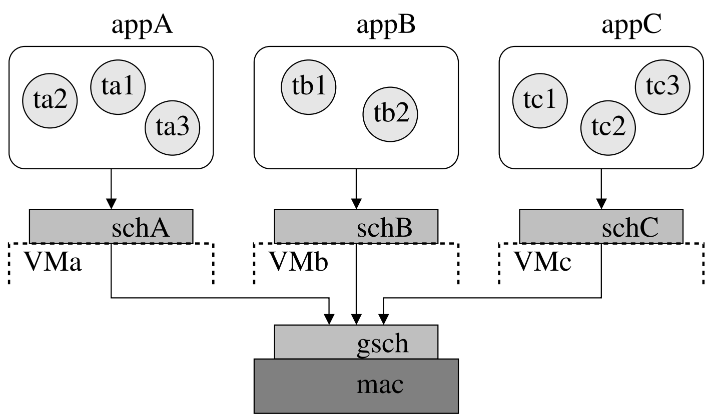
Motivations



Applications sharing the same machine. Issues:

- a misbehaviour in App a (running longer than expected, etc.) may cause misbehaviors on Apps b, c
- priority of application tasks needs to be comparable
  - may violate intellectual properties
  - when a new application joins, priorities of new tasks need to be assigned w.r.t. other tasks

Solution: hierarchical scheduling



- Machines becomes *virtual machines*
- Two schedulers:
  1. a *local scheduler* $\mathcal{S}_j$ schedules tasks of application j onto its own virtual machine
  2. a *global scheduler* $\mathcal{G}$ schedules virtual machines onto physical machines

Hierarchical scheduling: comments

- Benefits:
  - Applications are more portable (they just need a compliant virtual machine)
  - misbehaviours confined at app boundary
  - task prio meaningful only within app
- Comments
  - virtualization is the key technology that makes cloud computing feasible

Hierarchical real-time scheduling

Focus of next lectures:

how can real-time constraints of components be guaranteed over VMs?

1. Characteristics of tasks + local scheduler = time demand of a real-time applications
2. Available phisical resources + global scheduler = amount of time available to the VM (**main focus of next lectures**)
3. If

   time demand of the application "$\leq$" time provided by VM

   then the application is schedulable over the VM

Demand of applications

- We model an application by a set of n tasks (C_i, T_i, D_i)
  - C_i computation time
  - T_i period
  - D_i deadline
- Local scheduler $\mathcal{S}$ is assumed Fixed Priorities (FP)
  - other local schedulers are also possible

Time offered by a VM

- We restrict to single processor machines
  - depending on time, tomorrow, we will extend to virtual multiprocessors
- Virtual machines are not always available to applications
- Worst case for application is when the require the largest amount of time (Worst-Case Execution Time — WCET)
- Worst-case for a VM is when it provides the least amount of time to the application
  - It is needed an abstraction for the worst possible scenario for a VM

13 Supply upper/lower bound functions

Legal gloabl schedules of one VM

Given a global scheduling algorithm $\mathcal{G}$, we denote by $\mathcal{L}$ the set of all possible legal VM schedules $\mathcal{S}(t)$ as

$$\mathcal{L} = \{\mathcal{S} : \text{schedule } \mathcal{S} \text{ may be generated by } \mathcal{G}\},$$

We remind that if the number of physical machines is $|\mathcal{M}| = 1$

$$\mathcal{S} : \mathcal{T} \to \{0, 1\}$$

- $\mathcal{S}(t) = 1$, if the VM is scheduled by $\mathcal{G}$ onto a physical machine
- $\mathcal{S}(t) = 0$, if it is not

Example:

- $\mathcal{L}$ legal schedules of a VM implemented by a periodic servers with period P and budget Q;
- then $\mathcal{S} \in \mathcal{L}$ is such that...

Supply lower bound function

- The model of a virtual machine (VM) must capture their "not-fully-available" nature.

Definition 15 (informal)**.** We define the *supply lower bound function* $\mathsf{slbf}(t)$ of a VM as the minimum amount of resource available in any interval of length t, among any possible resource schedule.

Definition 16 (*supply lower bound function* of a VM [18, 14, 22])**.** We define the *supply lower bound function* $\mathsf{slbf}(t)$ of a VM as

$$\mathsf{slbf}(t) = \min_{t_0 \in \mathcal{T}, \mathcal{S} \in \mathcal{L}} \int_{t_0}^{t_0+t} \mathcal{S}(x)\, dx.$$

Supply upper bound function

Definition 17 (*supply upper bound function* of a VM)**.** We define the *supply upper bound function* $\mathsf{subf}(t)$ of a VM as

$$\mathsf{subf}(t) = \max_{t_0 \in \mathcal{T}, \mathcal{S} \in \mathcal{L}} \int_{t_0}^{t_0+t} \mathcal{S}(x)\, dx.$$

14 Supply function bounds: properties

Usage of supply bounds

Theorem 18 (Resource bounds)**.** *For any legal VM schedule $\mathcal{S} \in \mathcal{L}$, it is always*

$$\mathsf{slbf}(b-a) \leq \int_a^b \mathcal{S}(t)\, dt \leq \mathsf{subf}(b-a).$$

How long does it take to compute a work W over a VM?

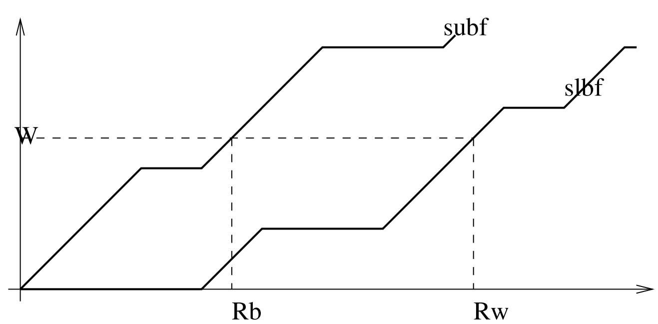
- *worst-case response time* $R_{\mathsf{w}}(W)$ of work W over a VM

$$R_{\mathsf{w}}(W) = \sup\{t : \mathsf{slbf}(t) < W\},$$

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

$$R_{\mathsf{b}}(W) = \inf\{t : \mathsf{subf}(t) \geq W\}.$$

Illustration of supply bounds and response-time bounds



Illustrate where the following sets are

- $\{t : \mathsf{slbf}(t) < W\}$
- $\{t : \mathsf{subf}(t) \geq W\}$

Example $\mathsf{slbf}(t)$, $\mathsf{subf}(t)$: static partition 1

- Let us assume the static VM schedule

$$\mathcal{S}(t) = \begin{cases} 1 & 0 \le (t \mod 4) < 1 \\ 0 & \text{otherwise} \end{cases}$$

- What are its $\mathsf{slbf}(t)$ and $\mathsf{subf}(t)$?
  - Remember: "any interval of length t" not necessarily $[0, t]$
- What are $R_{\mathsf{b}}(2)$ and $R_{\mathsf{w}}(2)$?

Example $\mathsf{slbf}(t)$, $\mathsf{subf}(t)$: static partition 2

- Let us assume the static VM schedule

$$\mathcal{S}(t) = \begin{cases} 1 & (t \mod 12) \in [2,3) \cup [5,7) \cup [10,12) \\ 0 & \text{otherwise} \end{cases}$$

- What are its $\mathsf{slbf}(t)$ and $\mathsf{subf}(t)$?
  - Remember: the interval with the minimum supply can start at different points for different t
- What is W such that $R_{\mathsf{w}}(W) - R_{\mathsf{b}}(W)$ over the VM, is maximized?
- "$R_{\mathsf{w}}(W) - R_{\mathsf{b}}(W)$" measures the uncertainty of the response time of a work W

Example $\mathsf{slbf}(t)$, $\mathsf{subf}(t)$: periodic server

- Let us assume that
  - the VM is implemented by a *periodic server*
  - the global scheduler guarantees a budget Q every period P
- What is its $\mathsf{slbf}(t)$? Remember: scenario of minimum possible supply must be assumed, among all legal schedules in $\mathcal{L}$

$$\mathsf{slbf}(t) = \begin{cases} 0 & t \in [0, P-Q] \\ (k-1)Q & t \in (kP-Q, (k+1)P-2Q] \\ t-(k+1)(P-Q) & \text{otherwise} \end{cases}$$

  with $k = \left\lceil \frac{t-(P-Q)}{P} \right\rceil$.

Properties of $\mathsf{slbf}(t)$

1. $\mathsf{slbf}(0) = 0$ (no resource in empty interval),
2. $\forall s \ge t \ge 0,\ \mathsf{slbf}(s) \ge \mathsf{slbf}(t)$ (non-decreasing)
3. $\forall s, t \ge 0,\ \mathsf{slbf}(s+t) \ge \mathsf{slbf}(s) + \mathsf{slbf}(t)$ (super-additivity)
4. $\forall s \ge t \ge 0,\ \mathsf{slbf}(s) - \mathsf{slbf}(t) \le s - t$ (Lipschitz continuous with factor 1):

Exploiting properties

Let us assume that at some instant t^* we know that

$$\mathsf{slbf}(t^*) \geq c^*, \tag{5}$$

From Lipschitz-continuity and (5), we find

$$\forall t \in [0, t^*] \quad \mathsf{slbf}(t) \geq \begin{cases} 0 & 0 \leq t \leq t^* - c^* \\ c^* + (t - t^*) & t^* - c^* < t \leq t^* \end{cases}$$

Let q and r be quotient and rest of the Euclidean division of any $t \geq 0$ by t^*, that is

$$t = qt^* + r, \qquad q \in \mathbb{N},\ r \in [0, t^*).$$

Then, for any $t \geq 0$, the following lower bound holds

$$\mathsf{slbf}(t) = \mathsf{slbf}(\underbrace{t^* + \cdots + t^*}_{q \text{ times}} + r) \geq \geq \underbrace{\mathsf{slbf}(t^*) + \cdots + \mathsf{slbf}(t^*)}_{q \text{ times}} + \mathsf{slbf}(r) \geq qc^* + \mathsf{slbf}(r).$$

15 Linear supply bounds

Linear lower bounds

- The exact supply functions may be too complicated to be handled.
- Supply lower bound function $\mathsf{slbf}(t)$ can be lower bounded by a linear function (**add drawing**):

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

$$\mathsf{lslbf}(t) = \max\{0, \alpha(t - \Delta)\}$$

with

$$\alpha \leq \lim_{t\to\infty} \frac{\mathsf{slbf}(t)}{t}, \qquad \Delta \geq \sup_{t \geq 0} \left\{ t - \frac{\mathsf{slbf}(t)}{\alpha} \right\}$$

although $\Delta > \ldots$ introduces a unnecessary pessimism.

Notice that the linear lower bound is not unique. For example $\mathsf{lslbf}(t) = 0$ is a valid (pessimistic) linear lower bound.

Linear lower bounds: properties

- α is often called *bandwidth* of the VM;
- Δ is often called *delay* of the VM, as the application may need to wait up to Δ before receiving the resource with rate α
  - $\Delta \geq \sup\{t : \mathsf{sbf}(t) = 0\}$ always.
- gain in simplicity: only two numbers (α, Δ) to abstract a VM;
- gain in generality: these two numbers can be computer for any global scheduler $\mathcal{A}$ (periodic, static time partition, etc.)
- loss in accuracy: there is some gap between the exact $\mathsf{slbf}(t)$ and its linear lower bound

lblsf: Example

1. What is the $\mathsf{lslbf}(t)$ of a periodic (P, Q) server?

$$\alpha = \frac{Q}{P}, \qquad \Delta = 2(P - Q)$$

   Given bandwidth α and delay Δ, a periodic server fulfilling (α, Δ) has the following periods

$$Q = \frac{\alpha\Delta}{2(1-\alpha)} \qquad P = \frac{\Delta}{2(1-\alpha)}$$

2. What is the $\mathsf{lslbf}(t)$ of

$$[1,2] \cup [3,6] \quad \text{with period } 6$$

$$\alpha = \frac{2}{3}, \qquad \Delta = 1.5 \ (> \text{longest idle time})$$

16 Hierarchical on multiprocessors

Main characteristic of multicore

Resource is available along *two dimensions*

- *horizontal* dimension over time
- *vertical* dimension over the number of processing units

In short:

- two machines of speed 0.5 and
- one machine of speed 1

are different.

Migration hypothesis:

1. migration/identical multiprocessor: the work can freely migrate over all CPUs and it takes the same amount of time
   - reasonable as first approximation
   - in reality migration has a non-uniform (cache hierarchies) cost

VM legal schedules

The schedule of a VM made by the global scheduler $\mathcal{G}$ is modeled by

$$\mathcal{S} : \mathcal{M} \times \mathcal{T} \to \{0, 1\}$$

with

- If $\mathcal{S}(k,t) = 1$ then the physical machine k is assigned to the VM at time t.
- If $\mathcal{S}(k,t) = 0$ then the physical machine k is not assigned to the VM at time t

Given a global scheduling algorithm $\mathcal{G}$, we denote by $\mathcal{L}$ the set of all possible legal VM schedules $\mathcal{S}(k,t)$ as

$$\mathcal{L} = \{\mathcal{S}(k,t) : \text{schedule } \mathcal{S} \text{ may be generated by } \mathcal{G}\},$$

Parallel supply lower bound function

Definition 19 (*parallel supply lower (bound) function* of a VM [5])**.** Given a VM, we define its *parallel supply lower bound function* $\mathsf{pslf}_k(t)$ with maximum parallelism k as

$$\mathsf{pslf}_k(t) = \min_{t_0 \in \mathcal{T}, \mathcal{S} \in \mathcal{L}} \int_{t_0}^{t_0+t} \min\left\{k, \sum_{\ell=1}^{m} \mathcal{S}(\ell, x)\right\} dx.$$

"minimum amount of resource made available by the VM in every interval of length t, with parallelism at most k". Notice that $\mathsf{pslf}_k(t)$ depends on

1. the length of the time interval t, **and**
2. the level of parallelism k,

to represent the bi-dimensional nature of the resource.

Parallel supply upper bound function

Definition 20 (*parallel supply upper (bound) function* of a VM)**.** Given a VM, we define its *parallel supply upper bound function* $\mathsf{psuf}_k(t)$ with maximum parallelism k as

$$\mathsf{psuf}_k(t) = \max_{t_0 \in \mathcal{T}, \mathcal{S} \in \mathcal{L}} \int_{t_0}^{t_0+t} \min\left\{k, \sum_{\ell=1}^{m} \mathcal{S}(\ell, x)\right\} dx.$$

Examples

1. $\mathsf{pslf}_k(t)$ of a VM with $\bar{m}$ dedicated processors?
2. $\mathsf{pslf}_k(t)$ of a VM which provides Q budget, every period P, with parallelism at most $\bar{m}$?

Properties

Same as sequential:

1. $\forall k,\ \mathsf{pslf}_k(0) = 0$ (no resource in empty interval),
2. $\forall k,\ \forall s \geq t \geq 0,\ \mathsf{pslf}_k(s) \geq \mathsf{pslf}_k(t)$ (non-decreasing)
3. $\forall k,\ \forall s, t \geq 0,\ \mathsf{pslf}_k(s+t) \geq \mathsf{pslf}_k(s) + \mathsf{pslf}_k(t)$ (super-additivity)
4. $\forall k,\ \forall s \geq t \geq 0,\ \mathsf{pslf}_k(s) - \mathsf{pslf}_k(t) \leq k(s-t)$ (Lipschitz continuous with factor k):

In addition:

1. from the definition, notice that $\mathsf{pslf}_0(t) = 0$
2. $\forall k,\ \forall t \geq 0,\ \mathsf{pslf}_k(t) \leq \mathsf{pslf}_{k+1}(t)$ (non-decr with k)
3. (def): $\bar{m} = \min\{k : \forall t,\ \mathsf{pslf}_k(t) = \mathsf{pslf}_{k+1}(t)\}$ is *max parallelism* of the VM;
4. $\forall k, \forall t \geq 0,\ \mathsf{pslf}_k(t) - \mathsf{pslf}_{k-1}(t) \geq \mathsf{pslf}_{k+1}(t) - \mathsf{pslf}_k(t)$ (concavity) [**explanation**]
5. $\mathsf{pslf}_{\bar{m}}(b-a) \leq \int_a^b \sum_{k=1}^m \mathcal{S}(k,t)\, dt \leq \mathsf{psuf}_{\bar{m}}(b-a)$

More properties

- $\mathsf{pslf}_1(t) \geq \max_{k \in \mathcal{M}} \mathsf{slbf}_k(t)$, with $\mathsf{slbf}_k(t)$ being the supply lower bound function of the k-th machine of the VM [**consider** t_0^* **and** $(s_1^*, \ldots, s_m^*)$ **which gives** $\mathsf{pslf}_1(t)$]
- If $\mathsf{pslf}_k(t^*) \geq c^*$, then

$$\forall t \in [0, t^*] \quad \mathsf{pslf}_k(t) \geq \begin{cases} 0 & 0 \leq t \leq t^* - \frac{c^*}{k} \\ c^* + k(t - t^*) & t^* - \frac{c^*}{k} < t \leq t^* \end{cases}$$

Let q and r be such that $t = qt^* + r, \qquad q \in \mathbb{N},\ r \in [0, t^*)$. Then, for any $t \geq 0$, the following lower bound holds

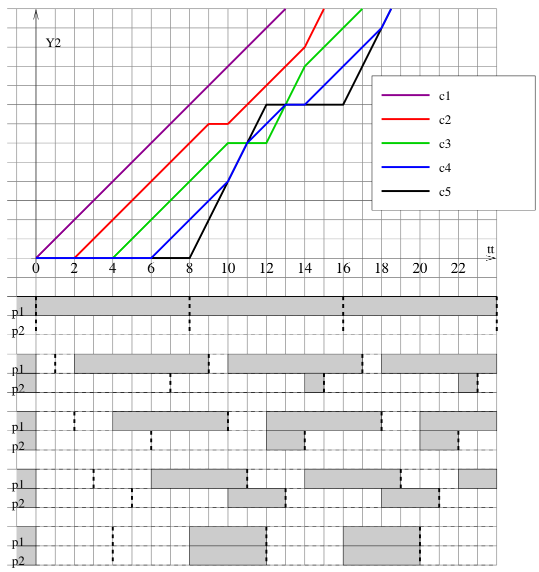
$$\mathsf{pslf}_k(t) = \mathsf{pslf}_k(\underbrace{t^* + \cdots + t^*}_{q \text{ times}} + r) \geq \geq \underbrace{\mathsf{pslf}_k(t^*) + \cdots + \mathsf{pslf}_k(t^*)}_{q \text{ times}} + \mathsf{pslf}_k(r) = qc^* + \mathsf{pslf}_k(r).$$

Examples

1. $\mathsf{pslf}_k(t)$ of a VM which provides $Q_1 = 4, Q_2 = 2$ budgets, on two CPUs, every period $P = 6$?

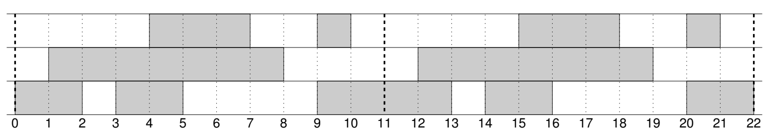
Is even budget always worst?

If periods are not synchronized [15]. . . ($P = 8$, $Q_1 + Q_2 = 8$)



Example of computation

Given the following resource schedule with period 11



what are:

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

17 Linear supply

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

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

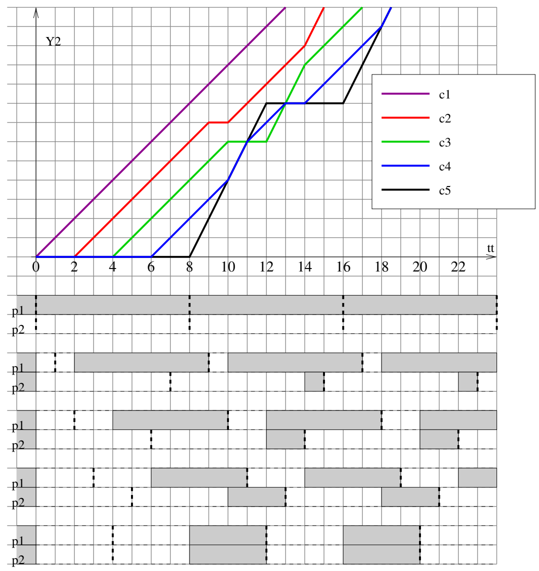
with

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

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

Linear supply: example

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



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 (for large t) 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?

18 Simple application model

Application model

Definition 21. 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 $(\leq D_i)$

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

Example 1: 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

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

Example 2: one rigid task
Given a rigid (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

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

19 Richer application model

Set of malleable tasks

Theorem 22 (FP-schedulability of malleable tasks)**.** *A set task of n constrained deadline (with $D_i \leq T_i$) malleable tasks is FP-schedulable over a VM with $\mathsf{pslf}_{\bar{m}}(t)$, if*

$$\forall i = 1, \ldots, n,\ \exists t \in \mathcal{P}_{i-1}(D_i), \quad C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq \mathsf{pslf}_{\bar{m}}(t)$$

with $\mathcal{P}_j(t)$ being a set recursively defined as

$$\begin{cases} \mathcal{P}_0(t) = \{t\} \\ \mathcal{P}_j(t) = \mathcal{P}_{j-1}\left(\left\lfloor \frac{t}{T_j} \right\rfloor T_j\right) \cup \mathcal{P}_{j-1}(t). \end{cases}$$

Set of non-malleable tasks

Theorem 23 (Schedulability of non-malleable tasks [15]). *A set of n constrained deadline non-malleable tasks is schedulable by the local scheduling algorithm $\mathcal{S}$ over the VM abstracted by $\{\mathsf{pslf}_k\}_{k=1}^{\bar{m}}$, if*

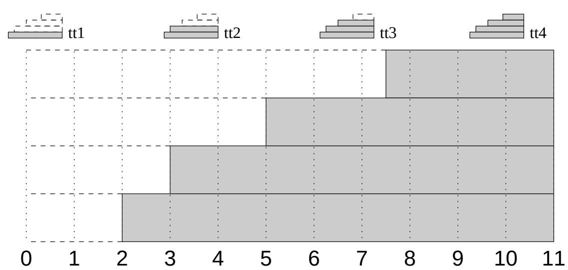
$$\bigwedge_{i=1,\ldots,n} \bigvee_{k=1\ldots,\bar{m}} k\, C_i + W_i^{\mathcal{S}} \leq \mathsf{pslf}_k(D_i),$$

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

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

- If local sched. algo. $\mathcal{S} = \mathsf{FP}$, 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 $N_{ji} = \left\lfloor \frac{D_i + D_j - C_j}{T_j} \right\rfloor$.

Proof sketch of Theorem 1/3

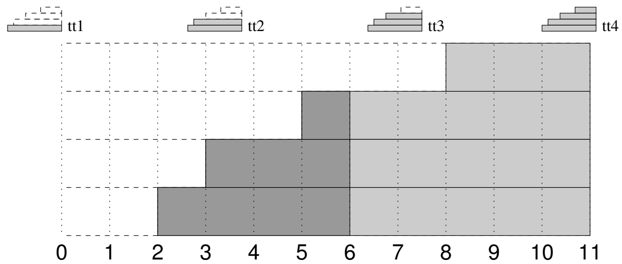


Let us assume

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

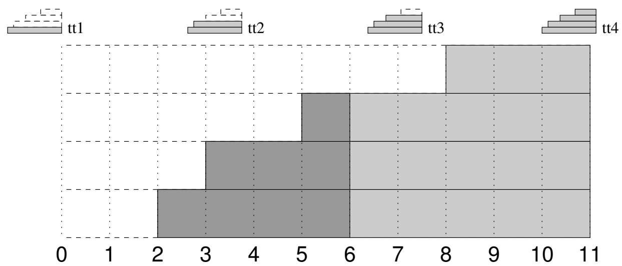
Proof sketch of Theorem 2/3

- 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 ${}^{\epsilon}\!/_{k}$
- 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 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 from $C_i + I_i \leq D_i$**]**

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 W_i is treated as *malleable* (it is assumed it can occupy any level of parallelism), while this is not the case.

20 Optimal design in Cyber-Physical Systems

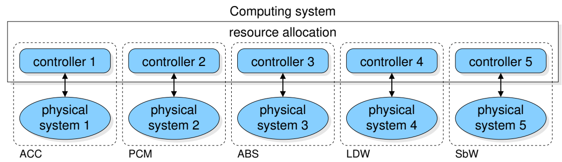
Motivation: CPS in embedded systems

The automotive case:

1. Large number of functions in cars: Antilock Brake System, Adaptive Cruise Control, Brake Assist System, Brake by Wire, Steer by Wire, Lane Departure Warning, ...
2. Number of functions expected to increase: Platooning, Highway Copilot, Collision Avoidance, Autonomous Driving, ...
3. Wiring is an issue: about 2 km in today's car (500 times the length). In Airbus A380 about 500 km of cables (7000 times its length): replacing copper with aluminum saved 30% of weight.

More than one controllers must co-exist over the same computing units

Controllers share resources



1. How to allocate the computing resource to the control tasks?
2. How to assign sampling frequency to controllers?
   - high frequency $\Rightarrow$ high control performance
   - low frequency $\Rightarrow$ high resource savings
3. How to assign priority to control tasks?
   - high priority $\Rightarrow$ low interference
   - low priority $\Rightarrow$ high interference (better for delay/jitter insensitive plants)

Optimal resource assignment

1. ϕ_i variables of the i-th controller. $\Phi = [\phi_1, \dots, \phi_n]$ all design variables.
   - sampling period T_i (or frequency $f_i = {}^1/_{T_i}$)
   - priority, if priority-based scheduler
2. Amount of consumed resource $\mathcal{R}(\Phi)$ of a given design Φ
   - image of $\mathcal{R}$ equipped with a partial order "$\preceq$" (speed of processor, amount of memory) to compare "more" or "less" resource
3. Cost $J_i(\Phi)$ of i-th controller
   - depends of the required performance of the plant (stability, etc)
   - does not depend only on ϕ_i
   - overall cost $J = \max_i J_i$ or $J = \sum_i J_i$
   - **cost-resource monotonicity** Let $\Phi = [\phi_1, \phi_2, \dots, \phi_n]$, $\Phi' = [\phi'_1, \phi_2, \dots, \phi_n]$. We can assume that $\mathcal{R}(\Phi') \succ \mathcal{R}(\Phi) \;\Rightarrow\; J_1(\Phi') < J_1(\Phi)$, otherwise Φ always better than Φ'

Problem formulation

- Given a bound B to the resource $\mathcal{R}(\Phi)$ consumed by the design variables Φ
- The goal is to solve the problem

$$\begin{aligned} &\underset{\Phi}{\mathsf{minimize}}\ J(\Phi) \\ &\mathsf{subject\ to}\ \mathcal{R}(\Phi) \preceq B \end{aligned}$$

- Possible a dual formulation with constraint on the cost and resource minimization.

21 Optimal design over FP

Problem formulation: periods are the variables

- The general problem formulation is

$$\begin{aligned} &\underset{\Phi}{\mathsf{minimize}}\ J(\Phi) \\ &\mathsf{subject\ to}\ \mathcal{R}(\Phi) \preceq B \end{aligned}$$

- If tasks are controllers, then
  - the variables become the sampling periods $(T_1, \ldots, T_n)$ of the controllers/tasks sharing the computing platform
  - the cost $J(T_1, \ldots, T_n)$ becomes the control cost (more details to come)
  - the constraint $\mathcal{R}(\Phi) \preceq B$ becomes the schedulability constraint of a set of tasks with periods $(T_1, \ldots, T_n)$
- The problem then is

$$\begin{aligned} &\underset{(T_1,\ldots,T_n)}{\mathsf{minimize}}\ J(T_1, \ldots, T_n) \\ &\mathsf{subject\ to}\ \text{“the task set with period } (T_1, \ldots, T_n) \text{ is sched.”} \end{aligned}$$

22 Cost of control

Cost of a task period?

- What is the impact of having the period of a controller short/long?
- In general, we may think that having a shorter task period is beneficial: we can run the application “faster”, etc
- The quality of the delivered performance is very related to the application domains. Examples:
  - accuracy of numerical routines;
  - quality of control action (more details next);
  - accuracy of multimedia players.
- In control system, there is a formal way to define the cost

Linear continuous-time systems: standard quadratic cost

In linear systems

$$\begin{cases} \dot{x} & = Ax + Bu \\ x(0) & = x_0 \end{cases}$$

with $x \in \mathbb{R}^n$ being the state of the plant and $u \in \mathbb{R}^m$ the control input, a classic cost to be minimized is

$$J = \int_0^\infty (x'(t)Qx(t) + u'(t)Ru(t))\, dt$$

with $Q \geq 0$, $R > 0$ weighting matrices.

Such a cost corresponds to require a convergent state x with a not too large control signal.

Standard quadratic cost: solution

The cost

$$J = \int_0^\infty (x'(t)Qx(t) + u'(t)Ru(t))\, dt$$

is minimized by setting the optimal control input u as state-feedback signal

$$u(t) = -R^{-1}B'Kx(t),$$

with K equal to the positive definite solution of the following *Algebraic Riccati Equation* (ARE)

$$KBR^{-1}B'K - A'K - KA - Q = 0. \tag{6}$$

The minimal cost is

$$J_\infty = x_0'Kx_0. \tag{7}$$

Quadratic cost: periodic sampling

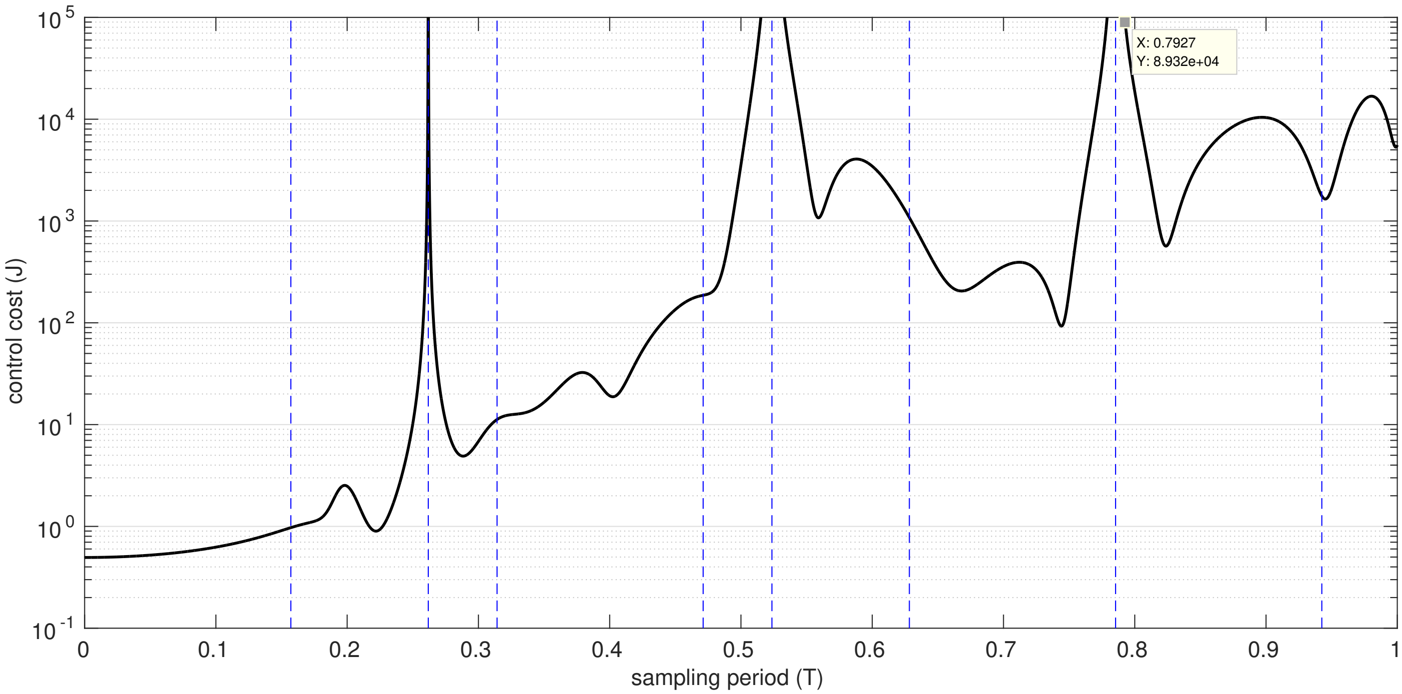
If the controller is not continuous-time, but it runs periodically with period T, then problem must be discretized with period T

$$\Phi(T) = e^{A\,T}, \qquad \Gamma(T) = \int_0^T e^{A(T-t)}\,dt\,B,$$
$$\bar{Q}(T) = \int_0^T \Phi'(t)Q\Phi(t)\,dt,$$
$$\bar{R}(T) = TR + \int_0^T \Gamma'(t)Q\Gamma(t)\,dt,$$
$$\bar{P}(T) = \int_0^T \Phi(t)'Q\Gamma(t)\,dt.$$

Let $\bar{K}(T)$ by the solution of *Discrete ARE* (DARE) of the discrete time systems $x_{k+1} = \Phi(T)x_k + \Gamma(T)u_k$ and weights $\bar{Q}(T)$, $\bar{R}(T)$, and $\bar{P}(T)$. Min cost is $J = x_0'\bar{K}(T)x_0$.

Cost $J(T)$ as function of T

Example of cost for a linear system

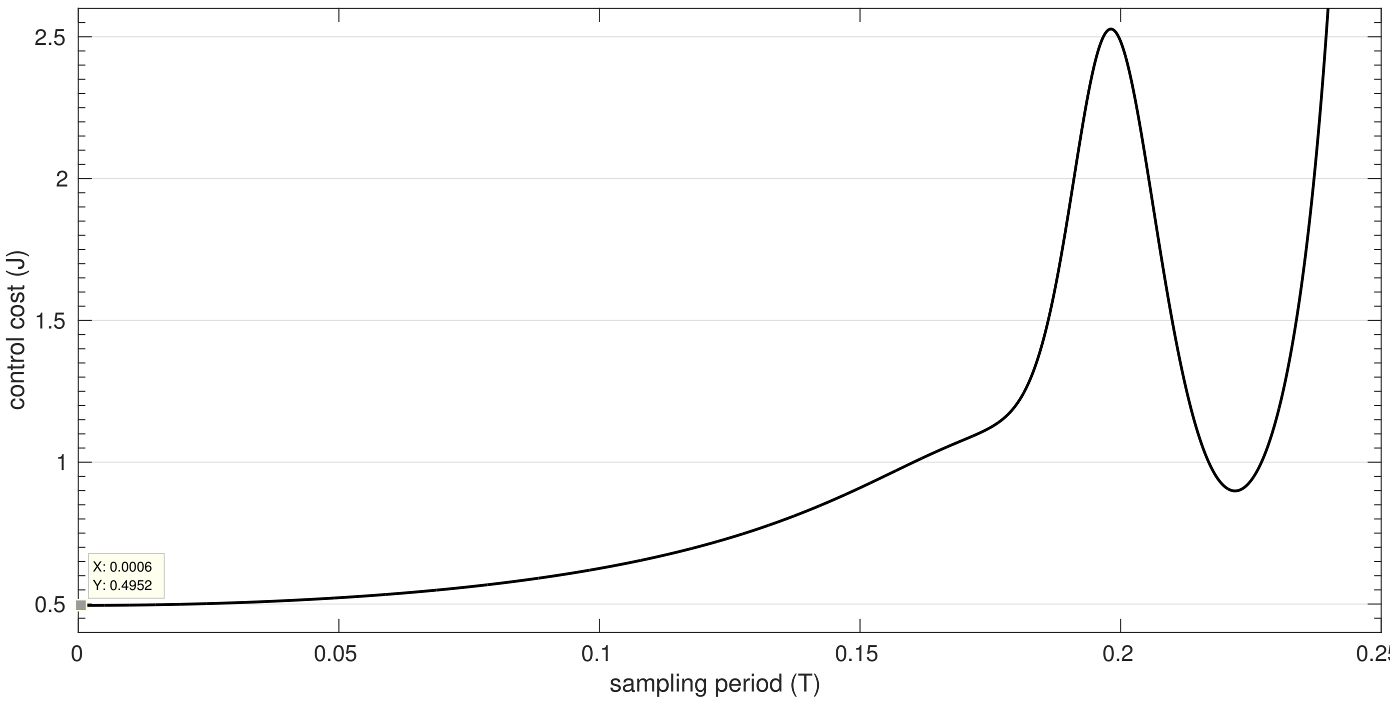


1. when $T \to 0$, $J(T)$ tends to the continuous-time cost J_∞
2. the cost is **not** always increasing with the sampling period
3. $J = \infty \quad \Rightarrow \quad T$ is a *pathological sampling period* (dashed lines): it exists λ_1, λ_2 eigenvalues of A, $\lambda_1 \neq \lambda_2$ with non-negative real part, such that $(\lambda_1 - \lambda_2)\frac{T}{2\pi} \in j\mathbb{Z}$

Cost J in neighborhood of $T \approx 0$

In a neighborhood of the origin ($T = 0$), the cost K can be approximated by a second order function with no first-order component [17]

$$J(T) = J_\infty + c_2T^2 + o(T^2)$$



Control cost with noise

- With noise, let the discretized dynamics be characterized by linear stochastic difference equation

$$x_{k+1} = \Phi(T)x_k + \Gamma(T)u_k + v(T)$$

  with $v(T)$ discrete-time white noise with variance $\sigma^2(T)$.

- In this case the cost is computed as

$$J = \lim_{t\to\infty} \frac{1}{t} E\left\{\int_0^t (x'(\tau)Qx(\tau) + u'(\tau)Ru(\tau))\, d\tau\right\}$$

  Factor $1/t$ is needed since $E\{\cdot\} \to \infty$.

- In a neighborhood of the origin $T \to 0$, $J(T)$ also has a non-zero first order component [10]

$$J(T) = J_\infty + c_1 T + o(T)$$

Cost of MPC

Model Predictive Control (MPC) is a popular control technique since:

- it allows the presence of constraints over the input and the state space;
- it guarantees stability [19];
- it finds the control signal that minimizes the control cost.

MPC can be tuned by two parameters:

- the sampling period T, and
- the prediction horizon H (the execution time C of the controller is then monotonic in H)

Recently [2], the cost $J(T, H)$ of MPC was formulated as function of the sampling period T and the prediction horizon H.

$J(T)$ is smooth only for small T (similarly as in linear systems).

Cost of control: summary

- Theory exists to compute the cost $J_i(T_i)$ of the i-th controller;
- In a neighborhood of $T_i = 0$ the cost $J_i(T_i)$ tends to the continuous-time cost:
  - as quadratic function if no noise is assumed
  - as linear function if noise is present
  - $J_i(T_i)$ can also be computed for MPC controllers
- cost $J_i(T_i)$ of a controller with sampling period T_i is not always increasing with T_i
- The overall cost is defined by a combination of individual costs J_i
  - when it is desired not to penalize any controller

$$J(T_1, \ldots, T_n) = \max_i J_i(T_i)$$

  - when it is acceptable to penalize some controller, if this leads to an overall benefit

$$J(T_1, \ldots, T_n) = \sum_i J_i(T_i)$$

23 FP: optimal period

Periods are design variables

- Solving the optimization problem

$$\begin{array}{c} \underset{(T_1,\ldots,T_n)}{\mathsf{minimize}}\ J(T_1,\ldots,T_n) \\ \mathsf{subject\ to}\ \forall i,\ \tau_i, \text{ with period } T_i, \text{ is sched. by RM} \end{array}$$

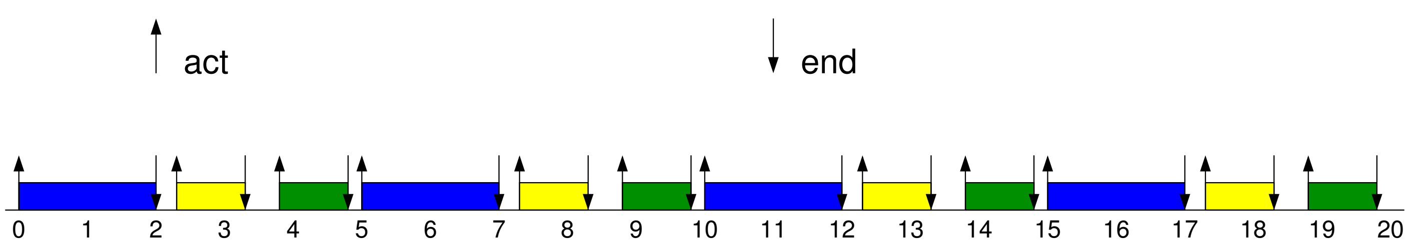
  requires the knowledge of the feasible region
- RM assigns priorities inversely proportional to task periods
- The geometry of the constraint "$\forall i,\ \tau_i$, with period T_i, is sched. by RM" is very complex (as we will see)
- Being very complex, some additional constraints may be added to simplify (and solve) the problem
  1. we may decide to impose all T_i equal to a common T, or
  2. we solve it on a linear subregion

First simple case: controllers have the same period

Let us denote by:

- C_i the worst-case execution time of the i-th controller,
- T_i the period of the i-th controller,
- $U_i = \frac{C_i}{T_i}$ the *utilization* of the controller, and
- $J_i(T_i)$ the cost of the i-th controller running with period T_i.

A simple solution is to set all periods T_i equal to a common schedule period T, and run all the n controllers in loop ($T = 5$ in schedule below).



Static schedule: solution

The optimal design solution is found by solving the problem

$$\begin{aligned} \underset{T}{\mathsf{minimize}} \quad & J(T) \\ \mathsf{subject\ to} \quad & \frac{\sum_i C_i}{T} \leq 1 \end{aligned}$$

which is a minimization problem over the only variable T with one constraint.

If the cost $J(T)$ is increasing over T, then

$$T^*(= T_1 = \cdots = T_n) = \sum_{i=1}^{n} C_i.$$

Pros:

- highly predictable schedule

Cons:

- timing of the schedule constrained by fastest dynamics (which is the one with $J_i(T_i)$ higher than the others)

Second simple case: linear subregion

If sampling periods T_i are not necessarily equal to each other

$$\begin{aligned} \underset{T_1,\ldots,T_n}{\mathsf{minimize}} \quad & J(T_1, \ldots, T_n) \\ \mathsf{subject\ to} \quad & \sum_{i=1}^{n} \frac{C_i}{T_i} \leq U_{\mathsf{LL}}(= n(\sqrt[n]{2} - 1)) \end{aligned}$$

If cost is sum of power functions (as in control systems in the smooth region)

$$J(T_1, \ldots, T_n) = \sum_{i=1}^{n} \alpha_i T_i^k$$

then exact solution can be found [21] by the Lagrange multipliers method, and it is

$$T_i^* = \frac{1}{U_{\mathsf{LL}}} \left(\frac{C_i}{k\,\alpha_i} \right)^{\frac{1}{k+1}} \sum_{j=1}^{n} \left(k\,\alpha_j C_j^k \right)^{\frac{1}{k+1}}$$

Notice that if the computation time of the controllers are the same (for example, if the controllers implements the same method, read a state of the same dimension, etc.) then

$$T_i^* \propto (k\alpha_i)^{-\frac{1}{k+1}},$$

that is, the larger α_i the smaller the corresponding T_i (not surprising).

In the special cases of linear ($k = 1$) or quadratic ($k = 2$, as it is in a neighborhood of $T = 0$) approximations, then we find

$$k = 1 \qquad T_i^* = \sqrt{\frac{C_i}{\alpha_i}} \sum_{j=1}^{n} \sqrt{\alpha_j C_j}$$

$$k = 2 \qquad T_i^* = \left(\frac{C_i}{\alpha_i} \right)^{\frac{1}{3}} \sum_{j=1}^{n} \left(\alpha_j C_j^2 \right)^{\frac{1}{3}}$$

Geometry of exact feasible periods

- Controllers/tasks have an execution time C_i
- Design variables Φ are task periods T_i
- Deadlines D_i constrains the feasible region $\mathcal{F}$

$$\mathcal{F}(C_1, \dots, C_n, D_1, \dots, D_n) = \{(T_1, \dots, T_n) : \forall i = 1, \dots, n \quad i\text{-th task with parameters } (C_i, T_i, D_i) \text{ is RM-schedulable}\}$$

($\mathcal{F}$ in short)

We remind that

$$\mathcal{F} \subseteq \left\{ (T_1, \dots, T_n) : \sum_i \frac{C_i}{T_i} \le 1 \right\}$$

because "total utilization not greater than 1" is necessary.

Problem formulation

Given:

- the cost $J(T_1, \dots, T_n)$, and
- the region of feasible periods $\mathcal{F}$

the optimal design problem is formulated as follows

$$\begin{aligned} \underset{T_1,\dots,T_n}{\mathsf{minimize}} \quad & J(T_1, \dots, T_n) \\ \text{subject to } & (T_1, \dots, T_n) \in \mathcal{F} \end{aligned}$$

What is the geometry of the FP-schedulable periods?

More convenient to perform a change of variables from task periods T_i to task activation *frequencies*

$$f_i = \frac{1}{T_i}$$

This way, the necessary condition $\sum_i U_i \le 1$ becomes a linear constraint $\sum_i C_i f_i \le 1$.

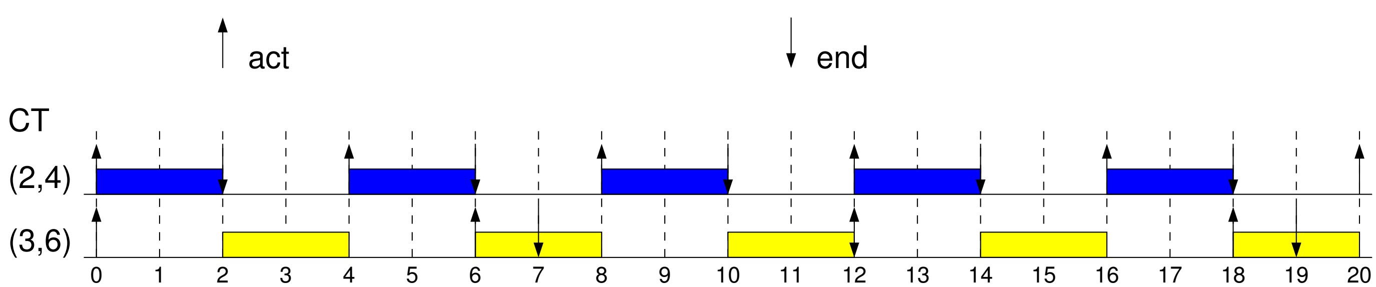
Understanding the feasible region

- To ease the presentation, we assume $D_i = T_i$

Theorem 24 ([3])**.** *The RM scheduling algorithm is* ***sustainable*** *w.r.t. task periods: if a period assignment* $(T'_1, \dots, T'_n)$ *is schedulable, then any period assignment* $(T''_1, \dots, T''_n)$*, with* $\forall i,\ T''_i \ge T'_i$ *is also schedulable.*

Theorem 25 ([12])**.** *If task periods are* ***harmonic****, that is* $\forall i, j, \frac{T_i}{T_j} \in \mathbb{N}$ *or* $\frac{T_j}{T_i} \in \mathbb{N}$*, then the task set is schedulable by RM* ***if and only if*** $\sum_{i=1}^{n} \frac{C_i}{T_i} \le 1$

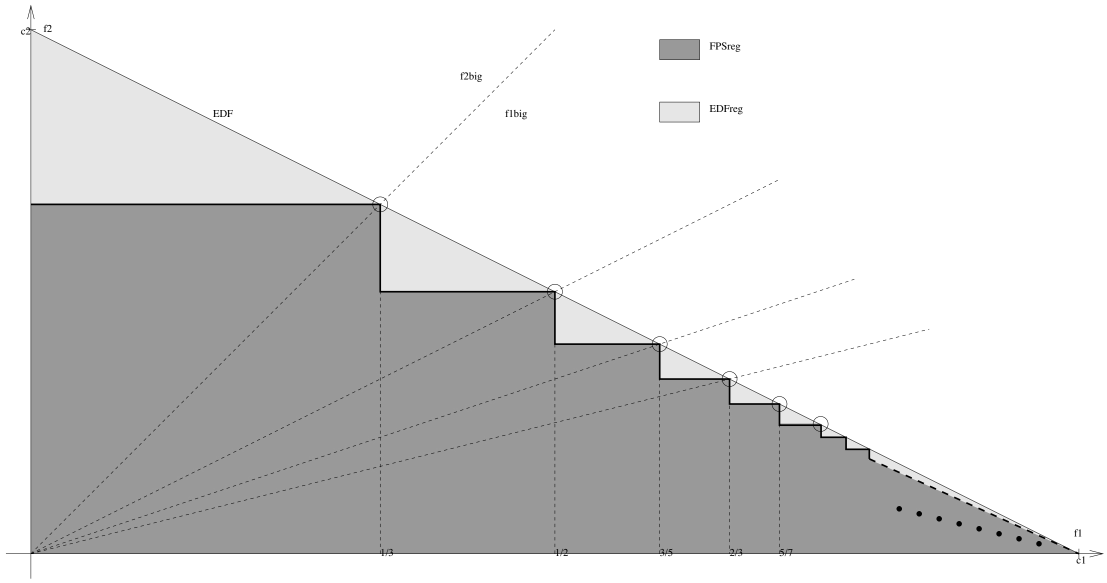
Not true in general (if not harmonic)



Space of RM-feasible frequencies

If $n = 2$ and $C_1 = 1$, $C_2 = 2$

- with harmonic periods, $\sum_i U_i \leq 1$ is an exact condition
- "sustainability" w.r.t. task periods

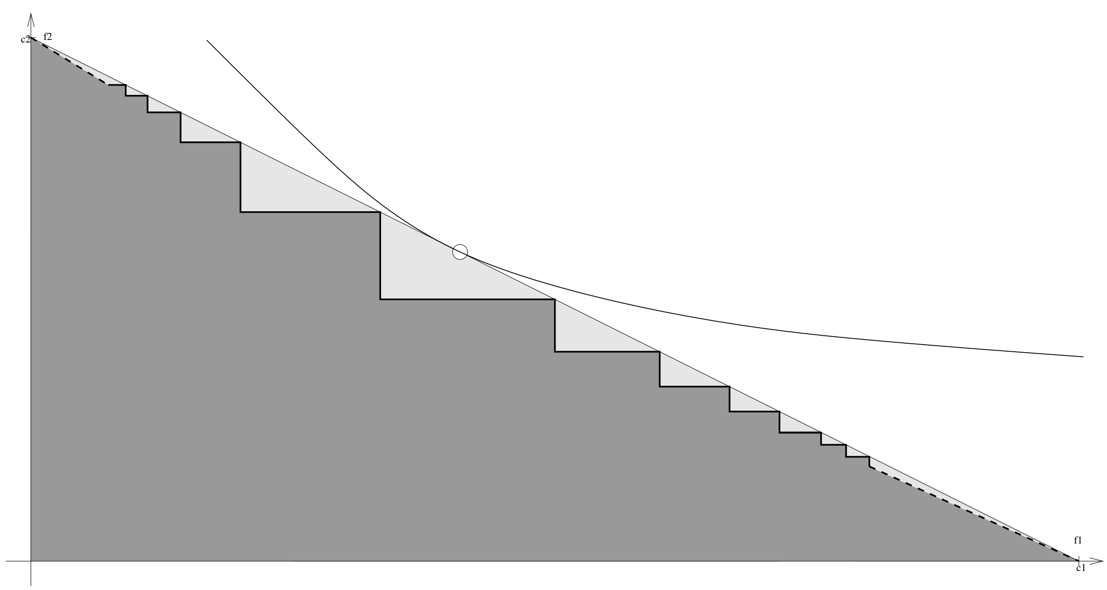


If $\frac{\partial J}{\partial f_i} \leq 0$ ($\Leftrightarrow \frac{\partial J}{\partial T_i} \geq 0$) all "vertices" are local optima.

If $n > 2$ feasible region is also formulated [7]

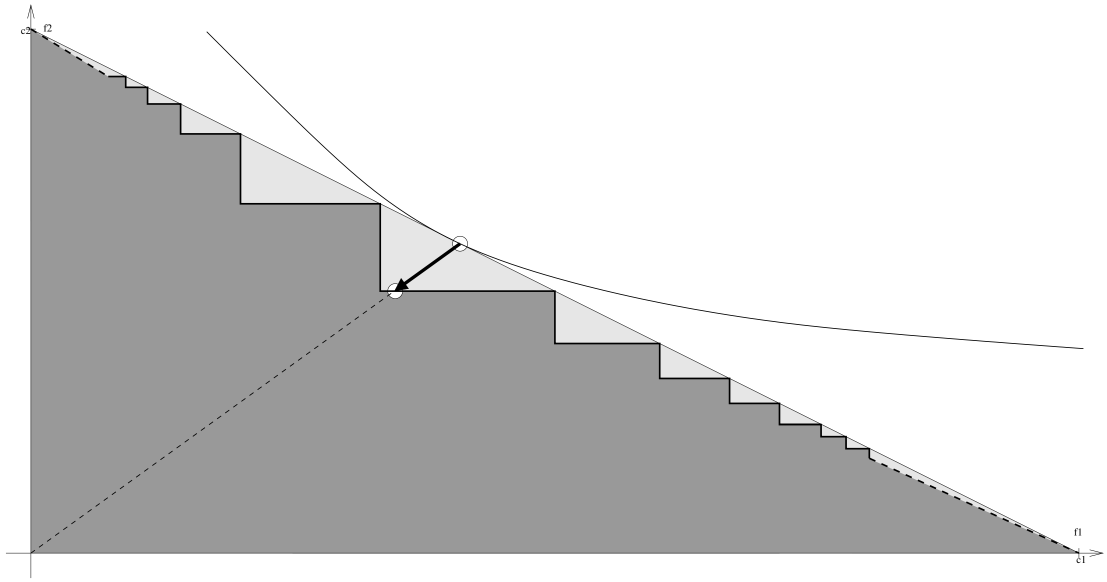
Effective heuristic

1. find a good starting point over a simple constraint (for example, by using $\sum_i U_i \leq 1$)



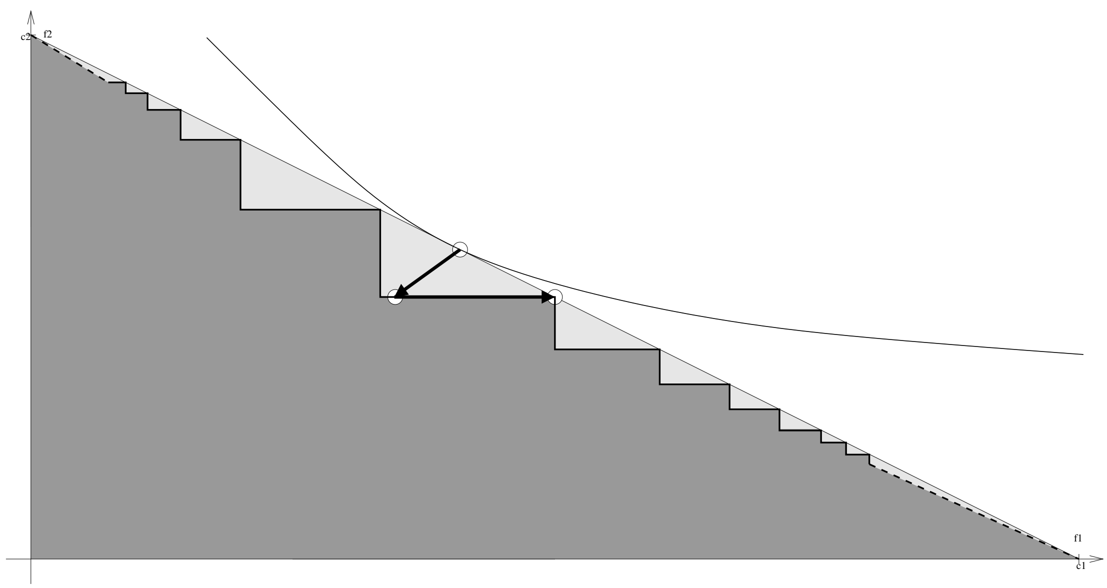
Effective heuristic

2. scale f_i down until RM boundary (sensitivity analysis [8]).



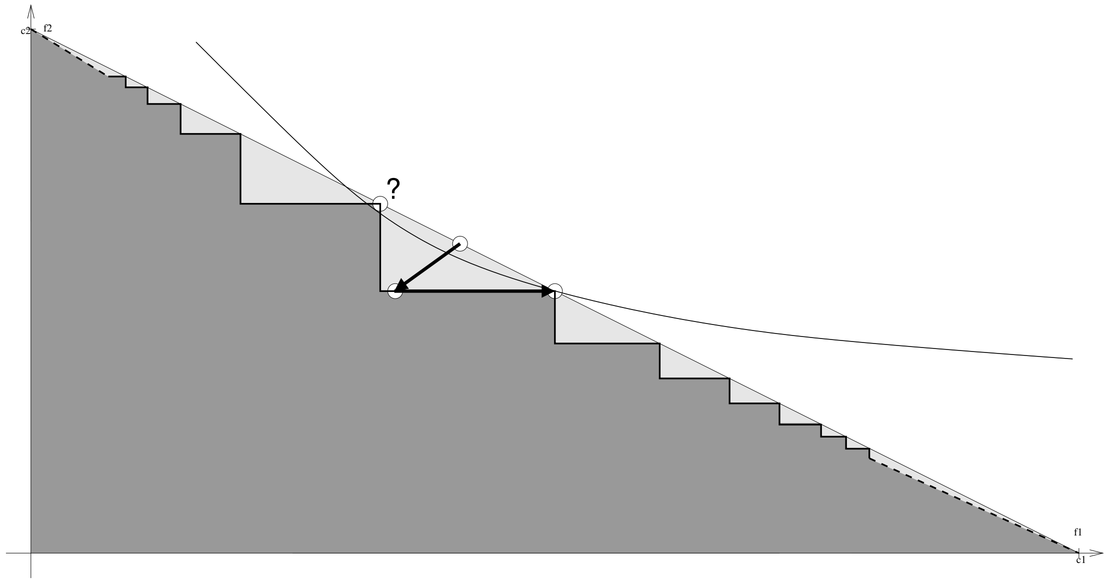
Effective heuristic

3. since $\frac{\partial J}{\partial f_i} \leq 0$, then by increasing task frequency ("Period sensitivity") we certainly cannot increase the cost J



Effective heuristic

4. No guarantee of optimality: feasible solution with lower cost may exist



Optimal search algorithm

Optimal search algorithm also exists [7]

1. Start from an initial RM-schedulable solution $\mathbf{f}^{\mathsf{first}}$ (such as the one found previously) and set it as current solution $\mathbf{f}^{\mathsf{cur}}$
2. Use a branch and bound algorithm to enumerate, and possibly prune, all vertices $\mathbf{f}$ with higher cost $J(\mathbf{f}) > J(\mathbf{f}^{\mathsf{first}})$
3. if a better solution is found, then update $\mathbf{f}^{\mathsf{cur}}$
4. when all vertices are checked or pruned, then $\mathbf{f}^{\mathsf{cur}}$ is the optimum

Conclusion: Fixed Priorities

1. Fixed Priorities is a possible scheduling algorithm for periodic tasks
2. Optimal priorities are Rate Monotonic (RM)
3. Sensitivity allows computing the admissible variations of the task set prameters
4. Optimal design is used to assign periods of tasks that share the same processor
5. In optimal design the cost of the periods drives the optimization
6. In control systems, there are methods to determine the of the control action
7. The feasible region is given by the RM-schedulability
   - exact approach: complex, lower cost
   - by adding constraints, the problem becomes simpler at the price of sub-optimality

References

[1] Karsten Albers and Frank Slomka. Efficient feasibility analysis for real-time systems with edf scheduling. In *Proceedings of Design, Automation and Test in Europe*, pages 492–497, 2005.

[2] Vincent Bachtiar, Eric C. Kerrigan, William H. Moase, and Chris Manzie. Smoothness properties of the mpc value function in open and closed-loop with respect to sampling time and prediction horizon. In *IEEE Asian Control Conference*, 2015.

[3] Sanjoy K. Baruah and Alan Burns. Sustainable scheduling analysis. In *Proceedings of the* 27^{rd} *IEEE Real-Time Systems Symposium*, pages 159–168, Rio de Janerio, Brazil, December 2006.

[4] Sanjoy K. Baruah, Aloysius K. Mok, and Louis E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the 11th IEEE Real-Time Systems Symposium*, pages 182–190, Lake Buena Vista (FL), USA, December 1990.

[5] Enrico Bini, Marko Bertogna, and Sanjoy Baruah. Virtual multiprocessor platforms: Specification and use. In *Proceedings of the 30th IEEE Real-Time Systems Symposium*, pages 437–446, Washinghton, DC, USA, December 2009.

[6] Enrico Bini and Giorgio C. Buttazzo. Schedulability analysis of periodic fixed priority systems. *IEEE Transactions on Computers*, 53(11):1462–1473, November 2004.

[7] Enrico Bini and Marco Di Natale. Optimal task rate selection in fixed priority systems. In *Proceedings of the 26th IEEE Real-Time Systems Symposium*, pages 399–409, Miami (FL), U.S.A., December 2005.

[8] Enrico Bini, Marco Di Natale, and Giorgio Buttazzo. Sensitivity analysis for fixed-priority real-time systems. *Real-Time Systems*, 39(1–3):5–30, 2008.

[9] UmaMaheswari C. Devi. An improved schedulability test for uniprocessor periodic task systems. In *Proceedings* 15^{th} *Euromicro Conference on Real-Time Systems*, pages 23–30, July 2003.

[10] Dan Henriksson and Anton Cervin. Optimal on-line sampling period assignment for real-time control tasks based on plant state information. In *Proceedings of the* 44^{th} *IEEE Conference on Decision and Control and European Control Conference*, Seville, Spain, December 2005.

[11] Mathai Joseph and Paritosh K. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, October 1986.

[12] Tei-Wei Kuo and Aloysius K. Mok. Load adjustment in adaptive real-time systems. In *Proceedings of the* 12^{th} *IEEE Real-Time Systems Symposium*, pages 160–170, San Antonio (TX), U.S.A., December 1991.

[13] John P. Lehoczky, Lui Sha, and Ye Ding. The rate-monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the* 10^{th} *IEEE Real-Time Systems Symposium*, pages 166–171, Santa Monica (CA), U.S.A., December 1989.

[14] Giuseppe Lipari and Enrico Bini. Resource partitioning among real-time applications. In *Proceedings of the 15-th Euromicro Conference on Real-Time Systems*, pages 151–158, Porto, Portugal, July 2003.

[15] Giuseppe Lipari and Enrico Bini. A framework for hierarchical scheduling on multiprocessors: from application requirements to run-time allocation. In *Proceedings of the 31st Real-Time Systems Symposium*, pages 249–258, San Diego, CA, USA, December 2010.

[16] Chung Laung Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.

[17] Stuart M. Melzer and Benjamin C. Kuo. Sampling period sensitivity of the optimal sampled data linear regulator. *Automatica*, 7:367–370, 1971.

[18] Aloysius K. Mok, Xiang Feng, and Deji Chen. Resource partition for real-time systems. In *Proceedings of the 7th IEEE Real-Time Technology and Applications Symposium*, pages 75–84, Taipei, Taiwan, May 2001.

[19] James B. Rawlings and Kenneth R. Muske. The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, 38(10):1512–1516, 1993.

[20] Ismael Ripoll, Alfons Crespo, and Aloysius K. Mok. Improvement in feasibility testing for real-time tasks. *Real-Time Systems*, 11(1):19–39, 1996.

[21] Danbing Seto, John P. Lehoczky, Lui Sha, and Kang G. Shin. On task schedulability in real-time control systems. In *Proceedings of the 17th IEEE Real-Time Systems Symposium*, pages 13–21, Washington, DC, USA, December 1996.

[22] Insik Shin and Insup Lee. Periodic resource model for compositional real-time guarantees. In *Proceedings of the* 24^{th} *Real-Time Systems Symposium*, pages 2–13, Cancun, Mexico, December 2003.

[23] Marco Spuri. Analysis of deadline scheduled real-time systems. Technical Report RR-2772, INRIA, France, January 1996.

[24] Fengxiang Zhang and Alan Burns. Schedulability analysis for real-time systems with edf scheduling. *IEEE Transactions on Computers*, 58(9):1250–1258, September 2009.