CHALMERS

Global scheduling

General characteristics:

- All ready tasks are kept in a common (global) queue
- When selected for execution, a task can be dispatched to an arbitrary processor, even after being preempted
- Task execution is assumed to be ”greedy”:
  - If higher-priority tasks occupy all processors, a lower-priority task cannot grab a processor until the execution of a higher-priority task is complete.

CHALMERS

Global scheduling

Advantages:

- Supported by most multiprocessor operating systems
  - Windows NT, Solaris, Linux, ...
- Effective utilization of processing resources
  - Unused processor time can easily be reclaimed

Disadvantages:

- Weak theoretical framework
  - Few results from the uniprocessor case can be used
- Poor resource utilization for hard timing constraints
  - No more than 50% resource utilization can be guaranteed
- Suffers from several scheduling anomalies
  - Sensitive to period adjustments

CHALMERS

Global scheduling

Complexity of schedulability analysis for global scheduling: (Leung & Whitehead, 1982)

> The problem of deciding if a task set is schedulable on m processors with respect to global scheduling is NP-complete in the strong sense.

Consequence:

There can only exist a pseudo-polynomial time algorithm for
(i) finding an optimal static priority assignment, or
(ii) feasibility testing

But not both at the same time!

CHALMERS

Global scheduling

The ”root of all evil” in global scheduling: (Liu, 1969)

> Few of the results obtained for a single processor generalize directly to the multiple processor case; bringing in additional processors adds a new dimension to the scheduling problem. The simple fact that *a task can use only one processor even when several processors are free at the same time* adds a surprising amount of difficulty to the scheduling of multiple processors.

Consequence:

We're in deep trouble! (Even p-fair scheduling suffers from this.)

CHALMERS

Weak theoretical framework

Underlying causes:

- Dhall's effect:
  - With RM, DM and EDF, some low-utilization task sets can be unschedulable regardless of how many processors are used.
- Dependence on relative priority ordering:
  - Changing the relative priority ordering among higher-priority tasks may affect schedulability for a lower-priority task.
- Hard-to-find critical instant:
  - A critical instant does not always occur when a task arrives at the same time as all its higher-priority tasks.

CHALMERS

(this page is intentionally blank)

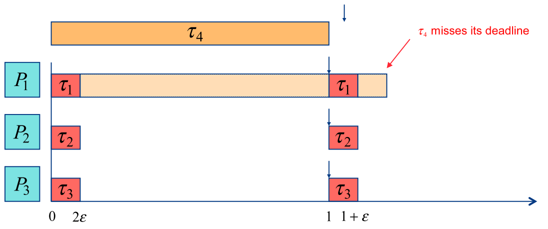
CHALMERS

Weak theoretical framework

Dhall's effect: (Dhall & Liu, 1978)

$\tau_1 = \{C_1 = 2\varepsilon, T_1 = 1\}$
$\tau_2 = \{C_2 = 2\varepsilon, T_2 = 1\}$
$\tau_3 = \{C_3 = 2\varepsilon, T_3 = 1\}$
$\tau_4 = \{C_4 = 1, T_4 = 1+\varepsilon\}$

(RM scheduling)



CHALMERS

Weak theoretical framework

Dhall's effect:

- Applies for (greedy) RM, DM and EDF scheduling
- Least utilization of unschedulable task sets can be arbitrarily close to 1 no matter how many processors are used.

$$U_{global} = m\frac{2\varepsilon}{1} + \frac{1}{1+\varepsilon} \rightarrow 1$$
when $\varepsilon \rightarrow 0$

Consequence:
New multiprocessor priority-assignment schemes are needed!

CHALMERS

Weak theoretical framework

Impact of relative priority ordering:

- The response time of a task depends on the relative priority ordering of the higher-priority tasks
- This property does not exist for a uniprocessor system
- This means that well-known uniprocessor methods for finding optimal priority assignments (e.g., Audsley, 1991) cannot be applied

Consequence:

New methods for constructing optimal multiprocessor priority assignments are needed!

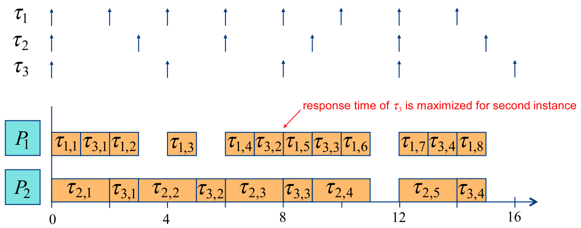
CHALMERS

Weak theoretical framework

Hard-to-find critical instant:

$\tau_1 = \{C_1 = 1, T_1 = 2\}$
$\tau_2 = \{C_2 = 2, T_2 = 3\}$
$\tau_3 = \{C_3 = 2, T_3 = 4\}$

(RM scheduling)



CHALMERS

Weak theoretical framework

Hard-to-find critical instant:

- A critical instant does not always occur when a task arrives at the same time as all its higher-priority tasks.
- Finding the critical instant is a very (NP-?) hard problem
- Note: recall that knowledge about the critical instant is a fundamental property in uniprocessor feasibility tests.

Consequence:

New methods for constructing effective multiprocessor feasibility tests are needed!

CHALMERS

Weak theoretical framework

Underlying causes:

- Dhall's effect:
  - With RM, DM and EDF, some low-utilization task sets can be unschedulable regardless of how many processors are used.
- Dependence on relative priority ordering:
  - Changing the relative priority ordering among higher-priority tasks may affect schedulability for a lower-priority task.
- Hard-to-find critical instant:
  - A critical instant does not always occur when a task arrives at the same time as all its higher-priority tasks.

New techniques for priority assignments and schedulability tests are needed!

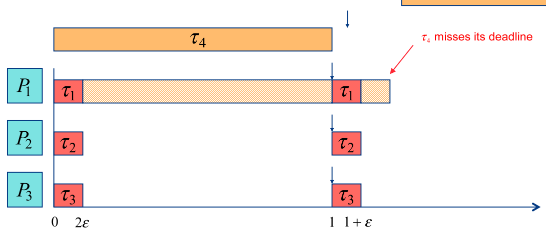
CHALMERS

Weak theoretical framework

Dhall's effect: (Dhall & Liu, 1978)

$\tau_1 = \{C_1 = 2\varepsilon, T_1 = 1\}$
$\tau_2 = \{C_2 = 2\varepsilon, T_2 = 1\}$
$\tau_3 = \{C_3 = 2\varepsilon, T_3 = 1\}$
$\tau_4 = \{C_4 = 1, T_4 = 1+\varepsilon\}$

(RM scheduling)

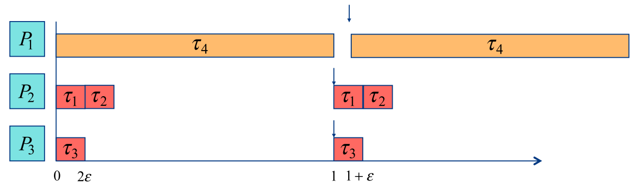


CHALMERS

New priority-assignment scheme

How to avoid Dhall's effect:

- Problem: RM, DM & EDF only account for task deadlines! Actual computation demands are not accounted for.
- Solution: Dhall's effect can easily be avoided by letting tasks with high utilization receive higher priority:



CHALMERS

New priority-assignment scheme

Algorithm RM-US[m/(3m-2)]:

(Andersson, Baruah & Jonsson, 2001)

- RM-US[m/(3m-2)] assigns (static) priorities to tasks according to the following rule:

  If $U_i > m/(3m-2)$ then τ_i has the highest priority (ties broken arbitrarily)

  If $U_i \leq m/(3m-2)$ then τ_i has RM priority

- Clearly, tasks with higher utilization, $U_i = C_i / T_i$, get higher priority.

CHALMERS

New priority-assignment scheme

RM-US[m/(3m-2)] example:

- As an example of the priorities assigned by RM-US[m/(3m-2)], consider the following task set to be scheduled on a system with 3 identical processors:

  $\tau_1 = \{C_1 = 1, T_1 = 7\}$ $\tau_2 = \{C_2 = 2, T_2 = 10\}$
  $\tau_3 = \{C_3 = 9, T_3 = 20\}$ $\tau_4 = \{C_4 = 11, T_4 = 22\}$
  $\tau_5 = \{C_5 = 2, T_5 = 25\}$

- The utilizations of these tasks are: 0.143, 0.2, 0.45, 0.5 and 0.08, respectively.

CHALMERS

New priority-assignment scheme

RM-US[m/(3m-2)] example:

- For $m = 3$:

  $m/(3m-2) = 3/7 \approx 0.4286$

- Hence, tasks τ_3 and τ_4 will be assigned higher priorities, and the remaining tasks will be assigned RM priorities.
- The possible priority assignments are therefore as follows (highest-priority task listed first):

  $\tau_3, \tau_4, \tau_1, \tau_2, \tau_5,$

  or

  $\tau_4, \tau_3, \tau_1, \tau_2, \tau_5,$

CHALMERS

New feasibility tests

Processor utilization analysis for RM-US[m/(3m-2)]:

(Andersson, Baruah & Jonsson, 2001)

- A sufficient condition for RM-US[m/(3m-2)] scheduling on m identical processors is

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq \frac{m^2}{3m-2}$$

- Question: does RM-US[m/(3m-2)] avoid Dhall's effect?

CHALMERS

New feasibility tests

Processor utilization analysis for RM-US[m/(3m-2)]:

- We observe that, regardless of the number of processors, the task set will always meet its deadlines as long as no more than one third of the processing capacity is used:

$$U_{RM-US[m/(3m-2)]} = \lim_{m \to \infty} \frac{m^2}{3m-2} = \frac{m}{3}$$

- RM-US[m/(3m-2)] thus avoids Dhall's effect since we can always add more processors if deadlines were missed.
- Note that this remedy was not possible with traditional RM.

CHALMERS

New feasibility tests

Response-time analysis for multiprocessors:

- Uses the same principle as the uniprocessor case, where the response time for a task τ_i consists of:

  C_i The task's uninterrupted execution time (WCET)

  I_i Interference from higher-priority tasks

$$R_i = C_i + I_i$$

- The difference is that the calculation of interference now has to account for the fact that higher-priority tasks can execute in parallel on the processors.

CHALMERS

New feasibility tests

Response-time analysis for multiprocessors:

- For the multiprocessor case, with n tasks and m processors, we observe two things:
  1. Interference can only occur when $n > m$.
  2. Interference can only affect tasks $\{\tau_k : k > m\}$ since the m highest-priority tasks will always execute in parallel without contention on the m processors.
- Consequently, interference of a task is a function of the execution overlap of its higher-priority tasks.

CHALMERS

New feasibility tests

Response-time analysis for multiprocessors:

- The following two observations give us the secret to analyzing the interference of a task:

  With respect to the execution overlap it can be shown that the interference is maximized when the higher-priority tasks completely overlap their execution.

  Compared to the uniprocessor case, one extra instance of each higher-priority task must be accounted for in the interference analysis.

CHALMERS

New feasibility tests

Response-time analysis for multiprocessors:

- The worst-case interference term is

$$I_i = \frac{1}{m} \sum_{\forall j \in hp(i)} \left(\left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j + C_j \right)$$

  where $hp(i)$ is the set of tasks with higher priority than τ_i.
- The worst-case response time for a task τ_i is thus:

$$R_i = C_i + \frac{1}{m} \sum_{\forall j \in hp(i)} \left(\left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j + C_j \right)$$

CHALMERS

New feasibility tests

Response-time analysis for multiprocessors:

- As before, an iterative approach can be used for finding the worst-case response time:

$$R_i^{n+1} = C_i + \frac{1}{m} \sum_{\forall j \in hp(i)} \left(\left\lceil \frac{R_i^n}{T_j} \right\rceil \cdot C_j + C_j \right)$$

- We now have a sufficient condition for static-priority scheduling on multiprocessors:

$$\forall i : R_i \le D_i$$

CHALMERS

Poor resource utilization

A fundamental limit: (Andersson, Baruah & Jonsson, 2001)

> The utilization guarantee bound for any static-priority multiprocessor scheduling algorithm cannot be higher than 1/2 of the capacity of the processors.

- This applies for all types of static-priority scheduling. That is, partitioned and global, greedy and p-fair scheduling.
- Hence, we can never expect to utilize more than half the processing capacity if hard timing constraints exist.
- The most resource-efficient multiprocessor real-time system is therefore one with a mix of soft and hard constraints.

CHALMERS

Scheduling anomalies

> **Scheduling anomaly**: A seemingly positive change in the system (reducing load or adding resources) causes a non-intuitive decrease in performance.

State-of-the-art :

- Uniprocessor systems:
  - Anomalies only found for non-preemptive scheduling (Mok, 2000)
- Multiprocessor systems:
  - Richard's anomalies for non-preemptive scheduling
  - Execution-time-based anomalies for preemptive scheduling
  - Period-based anomalies for preemptive scheduling

CHALMERS

Scheduling anomalies

Richard's anomalies: (Graham, 1969)

Assumptions:

- Non-preemptive scheduling
- Precedence constraints
- Restricted migration (individual task instances cannot migrate)
- Fixed execution times

Task completion times may increase as a result of:

- Changing the task priorities
- Increasing the number of processors
- Reducing task execution times
- Weakening the precedence constraints

CHALMERS

Scheduling anomalies

Execution-time-based anomalies: (Ha & Liu, 1994)

Assumptions:

- Preemptive scheduling
- Independent tasks
- Restricted migration (individual task instances cannot migrate)
- Fixed execution times

Task completion times may increase as a result of:

- Reducing task execution times

CHALMERS

Scheduling anomalies

Period-based anomalies: (Andersson & Jonsson, 2000)

Assumptions:
- Preemptive scheduling
- Independent tasks
- Full migration
- Fixed execution times

A task's completion time may increase as a result of:
- Increasing the period of a higher-priority task
- Increasing the period of the task itself

Note: increasing the periods is commonly used to reduce the load in feedback-control systems!

CHALMERS

Global scheduling

State-of-the-art in global scheduling:

- Static priorities:
  - The RM-US[m/(3m-2)] priority assignment scheme offers a way to circumvent Dhall's effect and a non-zero resource utilization guarantee bound of m/(3m-2) ≥ 33.3%.
  - In 2003, Baker generalized the RM-US results to DM.
- Dynamic priorities:
  - In 2002, Srinivasan & Baruah proposed the EDF-US[m/(2m-1)] scheme with a corresponding non-zero resource utilization guarantee bound of m/(2m-1) ≥ 50%.
- Optimal multiprocessor scheduling:
  - Using p-fair scheduling and dynamic priorities it is possible to achieve 100% resource utilization on a multiprocessor.