CHALMERS | UNIVERSITY OF GOTHENBURG

Parallel & Distributed Real-Time Systems

Lecture #8

Risat Pathan

Department of Computer Science and Engineering
Chalmers University of Technology

1

CHALMERS | UNIVERSITY OF GOTHENBURG

Multiprocessor scheduling

How are tasks assigned to processors?

- Static assignment
  - The processor(s) used for executing a task are determined before system is put in mission ("off-line")
  - Approaches: partitioned scheduling, guided search, non-guided search, ...
- Dynamic assignment
  - The processor(s) used for executing a task are determined during system operation "on-line"
  - Approach: global scheduling

2

CHALMERS | UNIVERSITY OF GOTHENBURG

Multiprocessor scheduling

How are tasks allowed to migrate?

- Partitioned scheduling (no migration!)
  - Each instance of a task must execute on the same processor
  - Equivalent to multiple uniprocessor systems!
- Guided search & non-guided techniques
  - Depending on migration constraints, a task may or may not execute on more than one processor
- Global scheduling (full migration!)
  - A task is allowed to execute on an arbitrary processor (sometimes even after being preempted)

3

CHALMERS | UNIVERSITY OF GOTHENBURG

Multiprocessor scheduling

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

The utilization guarantee bound for multiprocessor scheduling (strictly partitioned or strictly global) using static task priorities cannot be higher than 50% of the processing capacity.

One (older) approach to circumvent this limit is to use p-fair (priorities + time quanta) scheduling and dynamic task priorities.

Another (newer) approach to circumvent this limit is to split a certain number of tasks into two or more parts, and then run each part on a separate processor. The remaining tasks use partitioned scheduling.

4

CHALMERS | UNIVERSITY OF GOTHENBURG

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.

5

CHALMERS | UNIVERSITY OF GOTHENBURG

Global scheduling

Advantages:

- Supported by most multiprocessor operating systems
  - Windows 7, Mac OS X, Linux, ...
- Effective utilization of processing resources
  - Unused processor time can easily be reclaimed, for example when a task does not execute its full WCET.

Disadvantages:

Weak theoretical framework
- Few results from the uniprocessor case can be used

Suffers from several scheduling anomalies
- Sensitive to period adjustments

6

CHALMERS | UNIVERSITY OF GOTHENBURG

Global scheduling

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

The problem of deciding whether a task set (synchronous or asynchronous) 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!

7

CHALMERS | UNIVERSITY OF GOTHENBURG

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."

All schedulers that fulfill the 'no dynamic task parallelism' constraint suffers from this. (Even p-fair scheduling!)

8

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 | UNIVERSITY OF GOTHENBURG

(this page is intentionally blank)

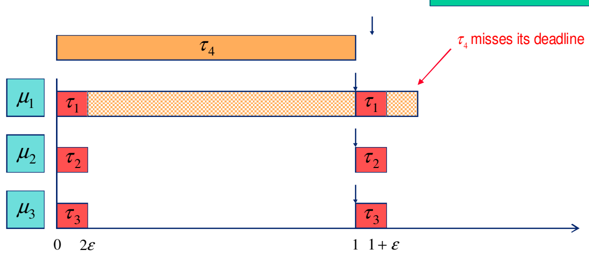
CHALMERS | UNIVERSITY OF GOTHENBURG

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 | UNIVERSITY OF GOTHENBURG

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 $\varepsilon \rightarrow 0$

Conclusion in year 2000: new priority-assignment policies for multiprocessors have to be developed!

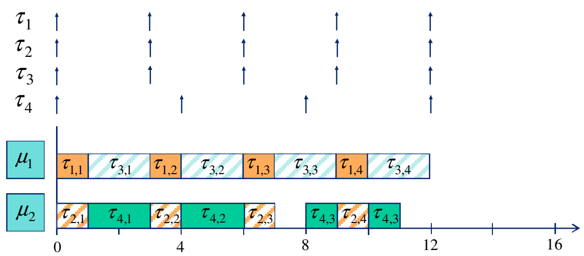
CHALMERS | UNIVERSITY OF GOTHENBURG

Weak theoretical framework

Impact of relative priority ordering:

$\tau_1 = \{C_1 = 1, T_1 = 3\}$
$\tau_2 = \{C_2 = 1, T_2 = 3\}$
$\tau_3 = \{C_3 = 2, T_3 = 3\}$
$\tau_4 = \{C_1 = 2, T_1 = 4\}$

RM scheduling
(priority order follows task index)



13

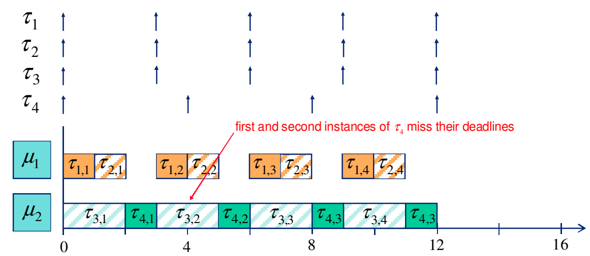
CHALMERS | UNIVERSITY OF GOTHENBURG

Weak theoretical framework

Impact of relative priority ordering:

$\tau_1 = \{C_1 = 1, T_1 = 3\}$
$\tau_2 = \{C_2 = 1, T_2 = 3\}$
$\tau_3 = \{C_3 = 2, T_3 = 3\}$
$\tau_4 = \{C_1 = 2, T_1 = 4\}$

RM scheduling
(priorities of τ_2 and τ_3 swapped)



14

CHALMERS | UNIVERSITY OF GOTHENBURG

Weak theoretical framework

Impact of relative priority ordering: (Andersson & Jonsson, 2000)

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 the OPA algorithm, which can be used on a uniprocessor for finding an optimal priority assignment, may not have that capability on a multiprocessor system.

Conclusion in year 2000: new methods for constructing optimal priority-assignment policies for multiprocessors may have to be developed!

15

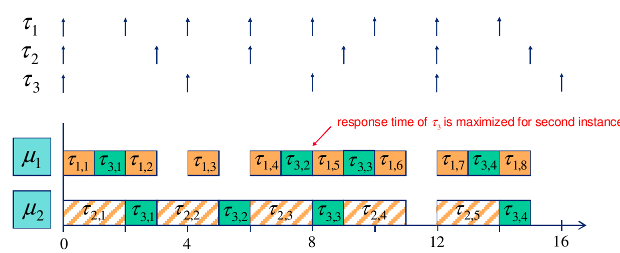
CHALMERS | UNIVERSITY OF GOTHENBURG

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



16

CHALMERS | UNIVERSITY OF GOTHENBURG

Weak theoretical framework

Hard-to-find critical instant: (Andersson & Jonsson, 2000)

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 the existence of a critical instant is a fundamental assumption in many efficient uniprocessor feasibility tests.

Conclusion in year 2000: new methods for constructing efficient feasibility tests for multiprocessors have to be developed!

17

CHALMERS | UNIVERSITY OF GOTHENBURG

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.

Conclusions in year 2000: new methods for priority assignments and schedulability tests for multiprocessors were needed!

18

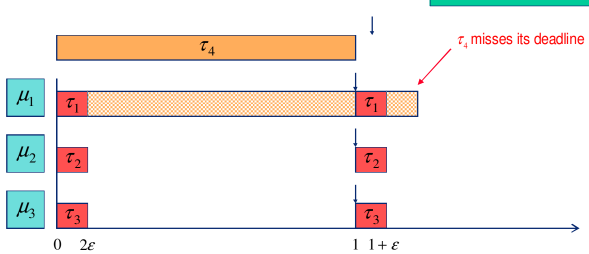
CHALMERS | UNIVERSITY OF GOTHENBURG

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



19

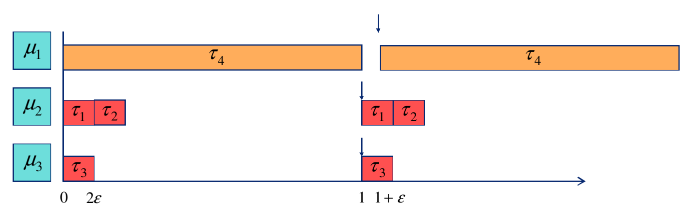
CHALMERS | UNIVERSITY OF GOTHENBURG

New priority-assignment scheme

How to avoid Dhall's effect:

Insight #1: RM, DM & EDF only account for task deadlines! Actual computation demands are not accounted for.

Insight #2: Dhall's effect can easily be avoided by letting tasks with high utilization receive higher priority:



20

New priority-assignment scheme

Scientific breakthrough: (Andersson, Baruah & Jonsson, 2001)

RM-US{m/(3m-2)}

- 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.

New feasibility test

Scientific breakthrough:

Guarantee bound analysis for RM-US{m/(3m-2)}:

- 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?

New feasibility test

Scientific breakthrough:

Guarantee bound 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 "pure" RM.

New feasibility test

Scientific breakthrough: (Andersson & Jonsson, 2000)

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

$$R_i = C_i + I_i$$

CHALMERS | UNIVERSITY OF GOTHENBURG

New feasibility test

Scientific breakthrough:

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 .

Note: The extra execution-time term introduced in this analysis is nowadays referred to as "carry-in" work.

25

CHALMERS | UNIVERSITY OF GOTHENBURG

New feasibility test

Scientific breakthrough:

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:

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

26

CHALMERS | UNIVERSITY OF GOTHENBURG

New optimality procedure

Scientific breakthrough: (Davis & Burns, 2009)

Conditions for OPA compatibility:

Condition 1: The schedulability of a task τ may, according to test S, depend on any independent properties of tasks with priorities higher than τ, but not on any properties of those tasks that depend on their relative priority ordering.

Condition 2: The schedulability of a task τ may, according to test S, depend on any independent properties of tasks with priorities lower than τ, but not on any properties of those tasks that depend on their relative priority ordering.

Condition 3: When the priorities of any two tasks of adjacent priority are swapped, the task being assigned the higher priority cannot become unschedulable according to test S, if it was previously schedulable at the lower priority.

27

CHALMERS | UNIVERSITY OF GOTHENBURG

New optimality procedure

Scientific breakthrough:

Conditions for OPA compatibility:

Task properties are referred to as independent if they have no dependency on the priority assigned to the task.
(e.g. WCET, period, deadline)

Task properties are referred to as dependent if they have a dependency on the priority assigned to the task.
(e.g. worst-case response time)

Feasibility tests which satisfy these conditions can be used together with the OPA algorithm to derive an optimal priority assignment on a multiprocessor system.

The multiprocessor response-time analysis shown earlier satisfies these conditions.

28

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

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

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

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 | UNIVERSITY OF GOTHENBURG

Global scheduling

State-of-the-art in global scheduling:

- Static priorities:
  - The SM-US{2/(3+√5)} priority-assignment policy has a guarantee bound of 38.2%. (Andersson, 2008)
- Dynamic priorities:
  - The EDF-US{m/(2m-1)} priority-assignment policy has a guarantee bound of 50%. (Srinivasan & Baruah, 2002)
- Task splitting:
  - The SPA2 task-splitting algorithm has a guarantee bound of 69.3% (c.f. the RM bound for uniprocessors). (Guan, et al., 2010)
- Optimal multiprocessor scheduling:
  - P-fair scheduling using dynamic priorities can achieve 100% resource utilization on a multiprocessor. (Baruah et al., 1995)

33

CHALMERS | UNIVERSITY OF GOTHENBURG

End of lecture #8

34