Real-Time Systems[1]

Sanjit Kumar Roy

Dept. of Information Technology

UPPSALA
UNIVERSITET

December 11, 2024

UPPSALA
UNIVERSITET

[1]Course ID: 1DT004 11210 and I 1DT063 11202 (HT2024)

Multiprocessor Scheduling - Day 2

UPPSALA UNIVERSITET

Outline

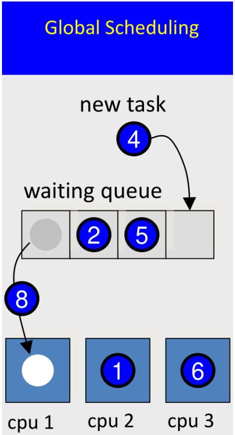
- Multiprocessor scheduling
  - Global scheduling
  - Partitioned scheduling
  - Semi-partitioned scheduling

UPPSALA UNIVERSITET

Multiprocessor Scheduling

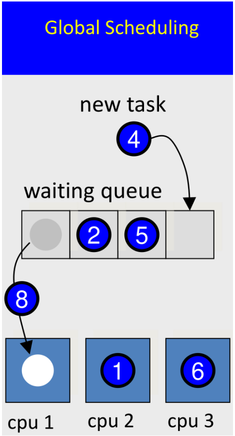
- Global scheduling (full migration)
  - A task is allowed to execute on an arbitrary processor (sometimes even after being preempted)
- Partitioned scheduling (no migration)
  - Each instance of a task must execute on the same processor
  - Equivalent to multiple uniprocessor systems!
- Semi-partitioned scheduling
  - Static task assignment
  - Each instance (or part of it) of a task is assigned to a fixed processor
  - Task instance or part of it may migrate

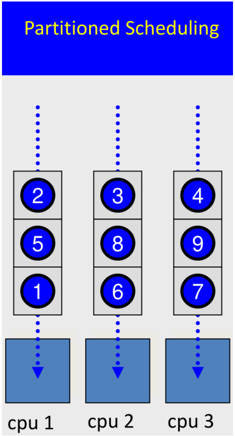
Multiprocessor Scheduling contd.



UPPSALA UNIVERSITET

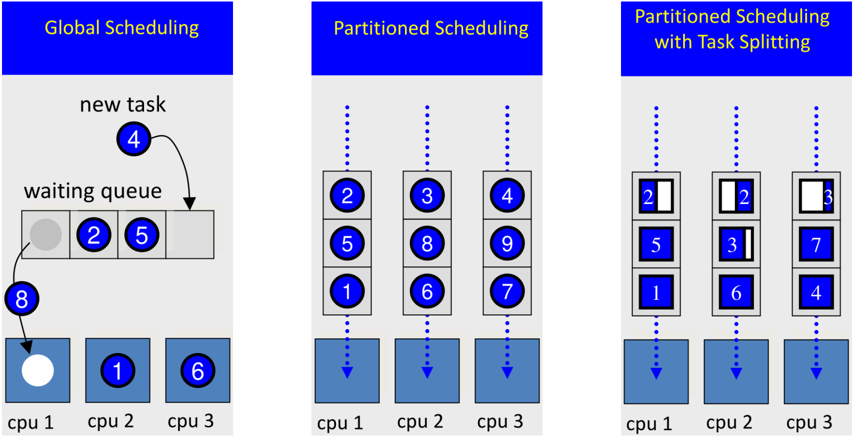
Multiprocessor Scheduling contd.





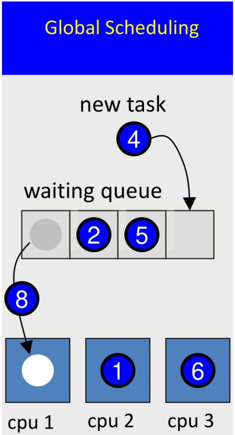
UPPSALA UNIVERSITET

Multiprocessor Scheduling contd.



UPPSALA UNIVERSITET

Global Scheduling



- All ready tasks are kept in a global queue
- When selected for execution, a task can be assigned to any processor, even after being preempted (task migration to another processor!)
  - Task migration may add overheads

UPPSALA UNIVERSITET

Global Scheduling contd.

Any algorithm for single processor scheduling may work, but schedulability analysis is non-trivial

- EDF (optimal for single processor)
  - Unfortunately EDF is not optimal for multiprocessors
    - Example: $\tau = \{(C_i, T_i)\} = \{(6, 12), (6, 12), (8, 13)\}$; Number of processors $m = 2$
  - No exact, but only sufficient schedulability test known
- Fixed Priority Scheduling (e.g. RM)
  - Difficult to find the optimal priority order
  - Difficult to check the schedulability
  - Suffer from the Dhall's effect

UPPSALA UNIVERSITET

Global Scheduling contd.

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

- With RM, DM and EDF, some low-utilization task sets can be unschedulable regardless of how many processors are used
  - Example: $\tau = \{(C_i, T_i)\} = \{(2\epsilon, 1), (2\epsilon, 1), (2\epsilon, 1), (1, 1+\epsilon)\}$
  - RM, DM, EDF can't schedule τ on any number of processors

UPPSALA UNIVERSITET

Global Scheduling contd.

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

- With RM, DM and EDF, some low-utilization task sets can be unschedulable regardless of how many processors are used
  - Example: $\tau = \{(C_i, T_i)\} = \{(2\epsilon, 1), (2\epsilon, 1), (2\epsilon, 1), (1, 1+\epsilon)\}$
  - RM, DM, EDF can't schedule τ on any number of processors
- Utilization:
  - Number of processors, m
  - A set of $m + 1$ tasks
    - m tasks each having utilization $2\epsilon/1$
    - One task has utilization $1/(1+\epsilon)$

UPPSALA UNIVERSITET

Global Scheduling contd.

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

- With RM, DM and EDF, some low-utilization task sets can be unschedulable regardless of how many processors are used
  - Example: $\tau = \{(C_i, T_i)\} = \{(2\epsilon, 1), (2\epsilon, 1), (2\epsilon, 1), (1, 1+\epsilon)\}$
  - RM, DM, EDF can't schedule τ on any number of processors
- Utilization:
  - Number of processors, m
  - A set of $m+1$ tasks
    - m tasks each having utilization $2\epsilon/1$
    - One task has utilization $1/(1+\epsilon)$

$$U = \frac{m \times 2\epsilon + \frac{1}{1+\epsilon}}{m} \to 0 \quad (\text{when } \epsilon \to 0 \text{ and } m \to \infty)$$

UPPSALA UNIVERSITET

Global Scheduling contd.

How can we avoid Dhall's effect:

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

UPPSALA UNIVERSITET

Global Scheduling contd.

How can we avoid Dhall's effect:

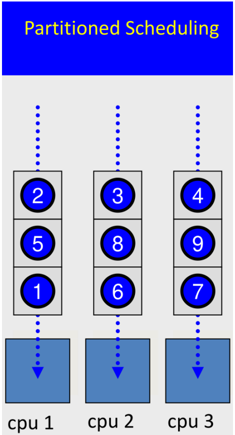
- RM, DM & EDF only account for task deadlines! Actual computation demands are not accounted for
- Dhall's effect can easily be avoided by letting tasks with high utilization receive higher priority:

New priority assignment scheme: (Andersson, Baruah & Jonsson, 2001)
$RM - US\{m/(3m-2)\}$

- $RM - US\{m/(3m-2)\}$ assign (static) priorities to tasks according to the following rule:
  - τ_i has highest priority if $U_i > m/(3m-2)$
  - τ_i has RM priority if $U_i \leq m/(3m-2)$
  - Here, $U_i = C_i/T_i$, and m is the number of processor

UPPSALA UNIVERSITET

Partitioned Scheduling



- Two steps:
  - Determine a mapping of tasks to processors
  - Perform run-time scheduling

UPPSALA UNIVERSITET

Partitioned Scheduling contd.

Bin-packing Algorithms:

- Pack items of different sizes into a minimum number of bins, each with a fixed capacity
  - Solutions (Heuristics): First Fit, Next Fit, Best Fit
- Application to multiprocessor systems:
  - Bins are represented by processors and objects by tasks
  - The decision whether a processor is “fully loaded” or not is derived from a utilization-based schedulability test

UPPSALA UNIVERSITET

Partitioned Scheduling contd.

Partitioned with EDF:

- Assign tasks to processors such that processor capacity does not exceeded (i.e. utilization bounded by 1)
- Schedule each processor using EDF

Partitioned Scheduling contd.

Partitioned with RM:

Rate-Monotonic First Fit (RMFF): (Dhall & Liu, 1978)

- First sort all tasks in non-decreasing order of their periods
- Task Assignment
  - Start from the highest priority task, and assign tasks to processors in the First Fit manner
  - A task τ_i can be assigned to a processor P_k if all the tasks assigned to P_k are RM-schedulable i.e.,
    - Total utilization of tasks assigned on P_k is bounded by $n(2^{1/n} - 1)$, where n is the number of tasks assigned
    - One may also use the precise test (response time analysis) to get a better assignment!
  - Add a new processor if needed for the RM-test

UPPSALA UNIVERSITET

Partitioned Scheduling contd.

Utilization bounded by 50%

- The Partitioning Problem is similar to Bin-packing problem (NP-hard)
- Limited Resource Usage, 50%
- Example:
  - Number of processors, m
  - A set of $m + 1$ tasks each having utilization $0.5 + \epsilon$

UPPSALA
UNIVERSITET

Partitioned Scheduling contd.

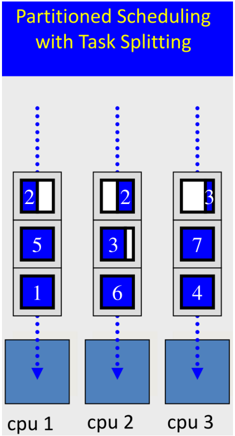
Utilization bounded by 50%

- The Partitioning Problem is similar to Bin-packing problem (NP-hard)
- Limited Resource Usage, 50%
- Example:
  - Number of processors, m
  - A set of $m+1$ tasks each having utilization $0.5+\epsilon$

$$U = \frac{(m+1)\times(0.5+\epsilon)}{m} \to 0.5 \quad (\text{when } \epsilon \to 0 \text{ and } m \to \infty)$$

UPPSALA UNIVERSITET

Semi-partitioned Scheduling



- Bin-packing with item splitting
- Resource can be “fully” (better) utilized

UPPSALA
UNIVERSITET

Semi-partitioned Scheduling contd.

Lakshmanan's Algorithm [ECRTS'09]:

- Sort all tasks in decreasing order of their utilization
- Pick up one processor, and assign as many tasks as possible from high utilization task to low utilization task
- If a task does not fit in the current processor then split the task into two
  - Assign one part of the task to the current processor
  - Pickup an empty processor, and assign another part of the task into it
- Utilization bound: 65%

Semi-partitioned Scheduling contd.

Breadth-First Partitioning Algorithms [RTAS 2010]

- Sort all tasks in increasing order of their priority
- Select the processor on which the assigned utilization is the lowest, and assign tasks from low priority to high priority
- If a task does not fit in the selected processor then split the task
- Utilization bound: 69%

UPPSALA UNIVERSITET