Partitioned scheduling of multimode multiprocessor real-time systems with temporal isolation

Joël Goossens[(1)], Pascal Richard[(2)]

[(1)] Université Libre de Bruxelles, Belgium
[(2)] LIAS, Université de Poitiers, France

Outlines

1. Problem statement
   - Mode changes in real-time systems
   - Transition Latency Delays
   - Task Allocation Problems
2. Offline Allocation Method : MILP
   - MILP main features
   - Numerical experiments
3. Online Allocation Method
4. Conclusion and Perspectives

System Model

- Sporadic tasks with Implicit Deadlines :
  $\tau = \{\tau_i(C_i, T_i), 1 \leq i \leq n\}$. Task Utilization : $U_i = C_i/T_i$
- Platform : *m* identical multiprocessor systems.
- Scheduler : partitioned EDF scheduling.

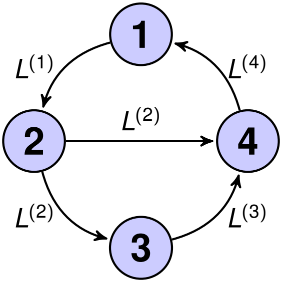
Many real-time applications have several operating modes :

- Aircraft : Take off / Flight / Landing
- Fault-Tolerance : Normal / Emergency / Fault-Recovery...

Mode changes in real-time systems

Graph of all possible Mode Transitions :

- Nodes represent Modes
- Edges represent Mode Transitions, labeled by worst-case transition delays.



Transition Scheduling Protocol

Every mode is initiated by an event : the Mode Change Request (MCR)

- Mode Independent (MI) tasks are run in every mode.
- Mode Dependent (MD) tasks are managed by a Transition Scheduling Protocol :
  - How Old Tasks (i.e., old mode) are stopped after the MCR,
  - How New Tasks are started.

Assumptions on the Task Set

Task set is assumed to be partitioned as follows :

- A Mode Dependent Task (MD) belongs to one, and only one, mode
- A Mode Independent Task (MI) is executed in every mode

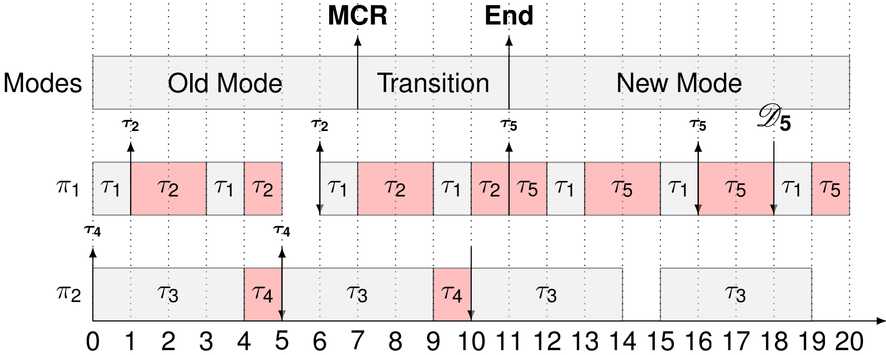
Synchronous Transition Scheduling Protocol

Assumptions on the Transition Scheduling Protocol :

- Old MD Tasks : every old job runs until completion after the MCR
- New MD tasks :
  - New MD tasks are launched only when every old MD task has been stopped (synchronous).
  - Temporal isolation of tasks running in different modes
  - Transition deadline : $\mathscr{D}_i$ (after the MCR)
- MI tasks : Mode Independent tasks continue their executions during the transition phase.

Example : Mode Change Request

- two processors : π_1, π_2 ; MI Tasks : $\tau_1(1,3), \tau_3(4,5)$;
- Old Tasks : $\tau_2(3,5), \tau_4(1,5)$; New Tasks : $\tau_5(3,5)$;



Transition Latency delay

Transition Latency Delay L : time interval between the MCR and the completion of Old jobs.

- Required for checking Transition Deadline ($\mathscr{D}_i$) of New MD tasks.
- Only an upper bound can be computed.

Property

In the given running mode, the transition latency delay L only depends on the tasks executed in the current mode.

Consequence : Task allocation problems can be solved mode by mode, independently.

Problem statements

Offline method for MD task allocation :

- Every MI task allocation is a priori known
- Allocation and Validation Problem : Compute the optimal MD task allocation so that the transition Latency Delay is minimized
- MD task allocations are stored in a Static Allocation Table.

Online method for task allocation :

- New tasks are allocated using First-Fit algorithm.
- Validation Problem : Algorithm for checking that task deadlines and transition deadlines are met.
- (main problem : how to compute a transition latency upper bound)

Problem statements

Offline method for MD task allocation :

- Every MI task allocation is a priori known
- Allocation and Validation Problem : Compute the optimal MD task allocation so that the transition Latency Delay is minimized
- MD task allocations are stored in a Static Allocation Table.

Online method for task allocation :

- New tasks are allocated using First-Fit algorithm.
- Validation Problem : Algorithm for checking that task deadlines and transition deadlines are met.
- (main problem : how to compute a transition latency upper bound)

Offline allocation (MILP)

MILP for allocating MD tasks in a given mode :

- Objective function :
  - Minimize the transition latency delay upper bound
- Decision variables :
  - binary variables : MD task allocations and disjunctive constraints
  - integer variables : number of MI jobs during the mode transition.
  - real variable : latency delay upper bound
- Constraints :
  - Every MD task is allocated
  - Allocations are feasible for each processor (utilization test)
  - Transition latency upper bounds

Upper bounds for a transition latency delay

The transition delay L is bounded either by :

- UB^1 : the greatest period among old tasks (since every allocation is feasible), or
- UB^2 : the longest synchronous busy period among all processors :
  - interference of MI tasks
  - completion of one job of every MD task

$L = min(\text{UB}^1, \text{UB}^2) \quad \Rightarrow \quad$ Disjunctive Constraints

MILP Formulation

The MILP looks like (Details are in the paper) :

$$
\begin{array}{ll}
\text{Minimize} \quad L & \\
\text{subjected to} & \\
\sum_{i=1}^{m} y_{ij} = 1 & j \in M \\
\sum_{\ell \in M} y_{i\ell} U_\ell + \sum_{\ell \in I_i} U_\ell \leq 1 & i = 1, \ldots, m \\
\sum_{\ell \in M} y_{i\ell} C_\ell + \sum_{\ell \in I_i} x_\ell C_\ell \leq x_j T_j & i = 1, \ldots, m; j \in I_i \\
\sum_{\ell \in M} y_{i\ell} C_\ell + \sum_{\ell \in I_i} x_\ell C_\ell \leq L + (1 - p_i)\mathrm{HV} & i = 1, \ldots, m \\
y_{ij} T_j \leq L + p_i \mathrm{HV} & i = 1, \ldots, m; j \in M \\
y_{ij} \in \{0, 1\} & i = 1, \ldots, m; j \in M \\
p_i \in \{0, 1\} & i = 1, \ldots, m \\
x_\ell \in \mathbb{N} & \ell \in I \\
L \in \mathbb{R} &
\end{array}
$$

Size of the MILP

For every mode is solved a MILP with :

- n : number of tasks ; m : number of processors ; M number of MD tasks
- Binary variables : $O(m \times M)$
- Integer variables : $O(M)$
- Real Variable : 1
- Constraints : $O(m \times n)$

Numerical experiments

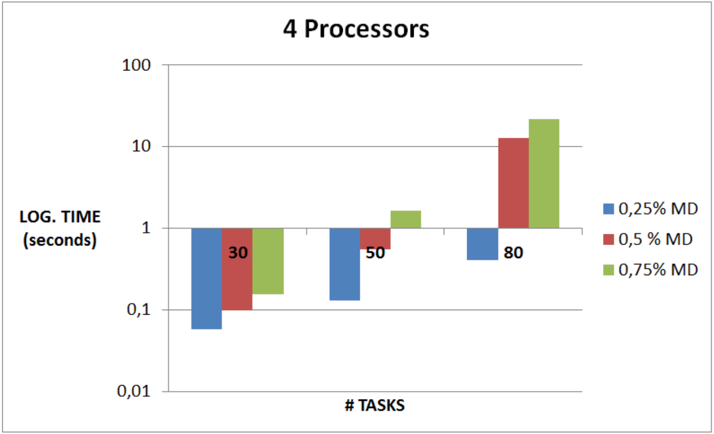
Task set synthesis :

- Task utilizations : Stafford's algorithm (RandFixedSum)
- Task periods : $\{5, 10, 15, 20, 50, 75, 100, 150, 500, 750, 1000\}$
- MI task allocation : Worst-Fit Decreasing (load balancing)

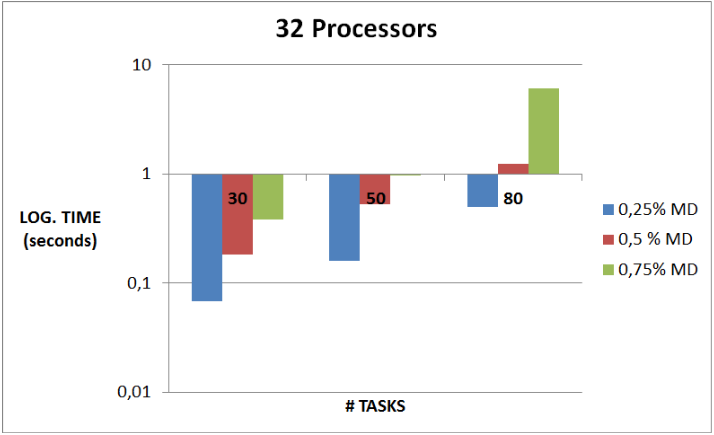
Numerical environment :

- Number of processors : $\{4, 16, 32\}$,
- Number of tasks : $\{30, 50, 80\}$,
- Percentage of Mode Dependent tasks in the task set : $\{25\%, 50\%, 75\%\}$.
- Platform utilization : $\{50\%, 66\%, 80\%\}$
- Replications : 100
- Time limit : 10 min (Gurobi MILP solver)

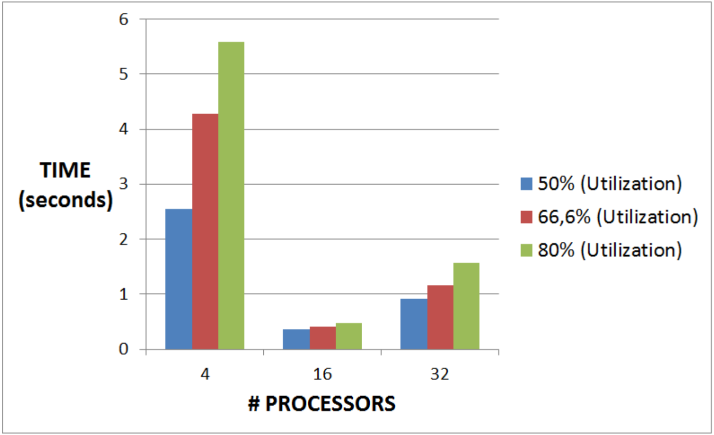
Results : 4 processors



Results : 32 processors



Results : Total Utilization



Validation of online MD task allocations

Online settings :

- MI tasks are statically allocated
- (online) First-Fit Allocation of MD tasks (at the beginning of the New Mode)

Validation problem for every mode :

- Task deadlines and Transition deadlines must be met
- Main problem : Transition Delay upper bound must cover all possible online allocations

Feasibility of online allocations

Schedulability condition for First-Fit allocation under EDF scheduling (Lopez et al. 2004).

$$\beta = \left\lfloor \frac{1}{U_{\max}} \right\rfloor \tag{1}$$

If the total utilization of tasks in the analyzed mode satisfies :

$$U_{\text{sum}} \leq \frac{\beta m + 1}{\beta + 1} \tag{2}$$

Then, First-Fit/EDF defines feasible schedules.

Bounding Transition Delays

Upon π_i, Transition delay upper bound L_i is defined by :

- execution time of every MD jobs (z_i), plus
- interference of MI tasks (fixed-point equation).

Problem

Which subset of MD task can be allocate to π_i in order to maximize the transition delay.

0-1 linear program for analysing π_i

compute which subset of MD task to allocate to each processor π_i, $1, \leq i \leq m$:

- Let I_i the set of MI task allocated to π_i and M be a set of MD tasks
- Binary variables : $y_\ell = 1$ if τ_ℓ is allocated to π_i, 0 otherwise.
- Maximize *the Transition Delay Upper Bound* z_i (i.e., longest sum of MD tasks processing times)

$$z_i = \sum_{\ell \in M} y_\ell C_\ell \tag{3}$$

- Subjected to the constraint *feasible allocation of MD tasks* :

$$\sum_{\ell \in M} y_\ell U_\ell \leq 1 - \sum_{\ell \in I_i} U_\ell \tag{4}$$

Validation algorithm for a given mode

- The valid Transition Delay Upper Bounds :

- ForEach π_i
  - $z_i := Solve(I_i, M)$ (i.e., knapsack problem related to π_i)
  - Compute smallest fixed-point of :

$$L_i := z_i + \sum_{\ell \in I_i} \left\lceil \frac{L_i}{T_\ell} \right\rceil C_\ell$$

- $L := \max_{i=1\cdots m}(L_i)$

- Check transition deadlines for every MD task.

Conclusion and Perspectives

Online/Offline Allocation methods for MultiMode Real-Time Systems :

- based synchronous protocol ensuring temporal isolation of running modes
- Allocation methods and Transition Latency Upper Bounds
- The Approach can be used for "real-world" systems

Perspectives : Extending this approach to

- Migrations of Mode Independent Tasks during Transition phase to allow higher utilization.
- Tasks with constrained and arbitrary deadlines