Technical Report

Partitioned Scheduling of Multimode Systems on Multiprocessor Platforms: when to do the Mode Transition?

José Marinho
Gurulingesh Raravi
Vincent Nélis
Stefan M. Petters

HURRAY-TR-110701
Version:
Date: 07-13-2011

Partitioned Scheduling of Multimode Systems on Multiprocessor Platforms: when to do the Mode Transition?

José Marinho, Gurulingesh Raravi, Vincent Nélis, Stefan M. Petters

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail:

http://www.hurray.isep.ipp.pt

Abstract

Systems composed of distinct operational modes are a common necessity for embedded applications with strict timing requirements. With the emergence of multi-core platforms protocols to handle these systems are required in order to provide this basic functionality.In this work a description on the problems of creating an effective mode-transition protocol are presented and it is proven that in some cases previous single-core protocols can not be extended to handle the mode-transition in multi-core.

Partitioned Scheduling of Multimode Systems on Multiprocessor Platforms: when to do the Mode Transition?

José Marinho, Gurulingesh Raravi, Vincent Nélis, Stefan M. Petters
CISTER-ISEP Research Center, Polytechnic Institute of Porto, Portugal.
{jmsm, ghri, nelis, smp}@isep.ipp.pt

Consider the problem of executing a multimode real-time system using partitioned scheduling on a multiprocessor platform. The model of computation is as follows (all technical terms are interpreted according to their usual definitions):

- **Platform:** The platform is composed of m identical processors $\{\pi_1, \ldots, \pi_m\}$.
- **System:** The system is composed of a task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$, where each task is assumed to be sporadic and implicit-deadline. The n tasks are divided into k subsets $\{\tau^1, \tau^2, \ldots, \tau^k\}$, where each subset τ^j represents the tasks that have to be executed while the system is running in mode M_j. The system can run in k different operational modes. These subsets τ^j are exhaustive but not mutually exclusive, i.e., $\tau^1 \cup \tau^2 \cup \ldots \cup \tau^k = \tau$ and $\exists j, i : \tau^i \cap \tau^j \neq \emptyset$. The tasks belonging to more than one mode are called *Mode-Independent* (MI) tasks.
- **Mode Transition:** At run-time, the application is either running in one of the modes (say M_i) or it is switching from one mode (i.e., the old mode) to another (i.e., new mode). A *mode transition phase* starts with a Mode Change Request (MCR) and ends when all the old mode tasks have completed the execution of their last released job (those old mode tasks do not release new jobs after an MCR) and all the new mode tasks have been activated, i.e. they have started to release jobs. Note that a mode-independent task that belongs to both the old and new mode should not be affected by the mode change in progress. Additionally no deadlines in the system may be violated.
- **Scheduling:** We consider partitioned scheduling, i.e., the task set τ^j of each mode M^j is partitioned into m subsets $\tau^{j,1}, \tau^{j,2}, \ldots, \tau^{j,m}$. Each partition $\tau^{j,\ell}$ is statically assigned to processor π_ℓ and is scheduled by preemptive EDF.
- **Assumptions:** Tasks are not allowed to migrate between processors during the execution of any mode. Furthermore, we assume that every subset of tasks $\tau^{j,\ell}$ $(\forall j, \ell)$ is EDF-schedulable.

Open problem: For any given transition phase, what is the earliest time-instant after the MCR at which all the new mode tasks can be safely activated?

Observation 1. *In order to preserve the schedulability of the system, it might be the case that some MI tasks have to migrate from one processor to another when a mode transition is initiated.*

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6
C_i	20	14	20	30	1	6
T_i	39.2	30	60	60	10	10
U_i	0.51	0.46	0.4	0.5	0.1	0.6
belongs to	τ^1	MI	MI	MI	τ^1	τ^2

Table I: A task set to illustrate the necessity of task migration across different modes.

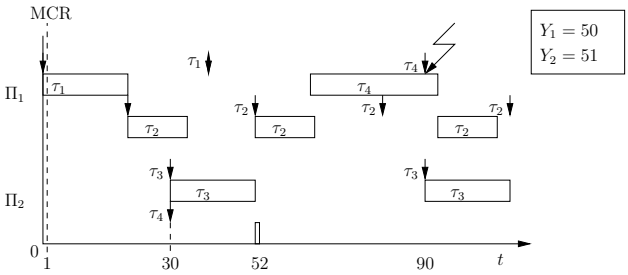

Figure 2: Schedule missing deadline with uniprocessor protocol adaptation

Consider a system with two modes M_1 and M_2, and six tasks $\{\tau_1, \ldots, \tau_6\}$. These tasks have to be scheduled on two processors π_1 and π_2, and their parameters are given in Table 1. The notation MI indicates that the task belongs to both modes M_1 and M_2. Thus we have $\tau^1 = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$ and $\tau^2 = \{\tau_2, \tau_3, \tau_4, \tau_6\}$. It can be shown that the only EDF-schedulable partitions for τ^1 is $\{\{\tau_1, \tau_2\}, \{\tau_3, \tau_4, \tau_5\}\}$. Similarly, the only EDF-schedulable partitions for τ^2 is $\{\{\tau_2, \tau_4\}, \{\tau_3, \tau_6\}\}$. That is, there is no EDF-schedulable partitions in which every MI task is assigned to the same processor in both modes M_1 and M_2, hence providing Observation 1.

Observation 2. *While it is commonly believed that uniprocessor scheduling techniques can be directly applied to partitioned multi-core scheduling, the presence of MI tasks may lead to counter-intuitive results.*

Consider the following uniprocessor result [1]: for any transition phase from mode M_i to mode M_j, the new-mode tasks can be safely activated at any time $t \geq t_{\text{MCR}} + Y_{i,j}$, where t_{MCR} is the time-instant of the last MCR and $Y_{i,j} \stackrel{\text{def}}{=} \sum_{\tau_\ell \in \tau^i} C_\ell + \sum_{\tau_\ell \in \tau^i \cap \tau^j} \left\lceil \frac{Y_{i,j}}{T_\ell} \right\rceil \times C_\ell$. A direct adaptation of the above protocol to partitioned multi-core would be: for any transition phase from mode M_i to mode M_j, the new-mode tasks can be safely activated at any time $t \geq t_{\text{MCR}} + \max_{k=1}^{m} \{Y_{i,j}^k\}$, where t_{MCR} is the

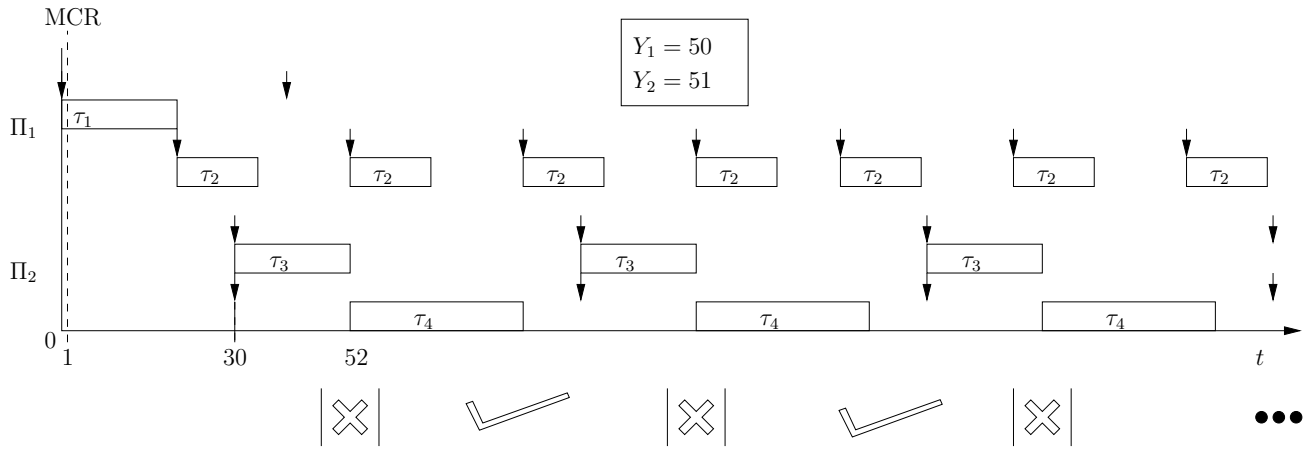

Figure 1: schedule of the two partitions with mode transition prolonged

instant of the last MCR and

$$Y_{i,j}^k \stackrel{\text{def}}{=} \sum_{\tau_\ell \in \tau^{i,k}} C_\ell + \sum_{\tau_\ell \in \tau^{i,k} \cap \tau^{j,k}} \left\lceil \frac{Y_{i,j}^k}{T_\ell} \right\rceil \times C_\ell.$$

However, the example using the system described in Figure 1 proves that this straight-forward extension of the protocol lead to deadline miss. In Figure 2 a deadline is missed at time $t = 90$ after the assumed end of the mode transition phase which completes at $t = 52$ since $t_{\text{MCR}} = 1$ and $\max_{k=1}^{m} \left\{ Y_{i,j}^k \right\} = 51$. It thus shows that waiting for the old mode workload to finish in every cores is not a sufficient condition to command the mode transition.

Observation 3. *There exist time intervals in which mode transition can be performed without missing any deadlines in the system. These intervals are separated by time intervals in which mode transition cannot be performed without missing a deadline.*

The extended schedule of the system discussed in Observation 1 is shown in Figure 1. We can observe from the schedule that the time intervals where it is safe to perform mode transition and time intervals where it is not safe to perform mode transition alternate. Hence, it is difficult to find at design time a time instant (using a uniprocessor multimode protocol) at which it is always safe to perform mode transition (as MCR is a run-time event).

Hence, from Observations 2 and 3, we can conclude that the uniprocessor multimode scheduling techniques cannot be directly applied to a multiprocessor multimode partitioned scheduling scenario.

ACKNOLEDGEMENTS

This work was supported by the RePoMuC project, ref. FCOMP-01-0124-FEDER-015050, funded by FEDER funds through COMPETE (POFC - Operational Programme Thematic Factors of Competitiveness) and by National Funds (PT) through FCT - Portuguese Foundation for Science and Technology and the RECOMP the project, funded by National Funds, through the FCT, under grant ref.- ARTEMIS/0202/2009 , as well as by the ARTEMIS Joint Undertaking, under grant agreement Grant nr. 100202.

REFERENCES

[1] Jorge Real, Alfons Crespo: Mode Change Protocols for Real-Time Systems: A Survey and a New Proposal. Real-Time Systems 26(2): 161–197 (2004).