Improvement of schedulability bound by task splitting in partitioning scheduling

Frédéric Fauberteau, Serge Midonnet, Laurent George

Improvement of schedulability bound by task splitting in partitioning scheduling

Frédéric Fauberteau
Université Paris-Est
LIGM, UMR CNRS 8049
Email: fauberte@univ-mlv.fr

Serge Midonnet
Université Paris-Est
LIGM, UMR CNRS 8049
Email: midonnet@univ-mlv.fr

Laurent George
ECE, LACSC
37, quai de Grenelle,
75015 Paris, France
Email: lgeorge@ieee.org

Abstract

We focus on the class of static-priority partitioning scheduling algorithm on multiprocessor. We are interested in improving the schedulability of these algorithms by splitting the tasks which cannot be successfully allocated on processors.

1. Context

Unless $P = NP$, no polynomial time algorithm exists to solve the MULTIPROCESSOR-TASK-PARTITIONING problem. Indeed, this problem can be transformed from BIN-PACKING problem which is NP-hard in the strong sense. Fortunately, there are several heuristics which solve BIN-PACKING problem and they may be adapted to produce partitioned schedulings. For instance, the algorithm FBB-FDD is based on *First-Fit Decreasing* [1] and the algorithm RM-DU-NFS is based on *Next-Fit* [2].

We are interested in proposing a partitioning algorithm which is robust to task Worst Case Execution Time (WCET) overruns faults with the same objective as in [3]. This partitioning tends to maximize the capacity of the system to handle the WCET overruns by offering the maximum *Allowance* to each faulty task of the system. The *Allowance* of a task is the execution duration which can be added to its WCET such that all the deadlines in the system are met [4].

We have shown in [5] that the *Worst-Fit* heuristic offers good results in order to maximize the robustness of a partitioned system. *Worst-Fit* selects the least loaded processor (in terms of utilization) which can accept a task. Unfortunately, the performances of *Worst-Fit* in terms of schedulability are less efficient than those of the heuristics widely used in partitioning algorithms (*First-Fit Decreasing*/*Best-Fit*/*Next-Fit*). In order to improve the schedulability bound of *Worst-Fit* and potentially those of the other heuristics, we propose to split the tasks which cannot be allocated one a single processor. Contrary to the portioned scheduling approach proposed in [6] and [7], we don't migrate a job from a processor to another during its execution but we allocate jobs on different processors such that the migrations occur at job boundaries [8].

2. Task splitting approach

Description. Let Π be a multiprocessor composed by m identical processors denoted by $\Pi = \{\pi_1, \ldots, \pi_m\}$. Let τ be a set of tasks made of n periodic (or sporadic) real-time tasks denoted $\tau = \{\tau_1, \ldots, \tau_n\}$. Each task is characterized by its WCET, its deadline and its period (or minimum interarrival time). We consider a partitioning algorithm denoted A which allocates the tasks of τ on Π. A is designed as follows:

- τ is sorted according to a decreasing order policy (for instance *Decreasing Utilization*),
- a heuristic chooses the processor on which a given task should be allocated,
- a schedulability test is used to decide whether the task can be allocated on the processor chosen by the heuristic.

The resulting algorithm A produces a partition of τ on Π. If A fails to allocate the task τ_i, the set of pending tasks $\{\tau_i, \ldots, \tau_n\}$ is unallocated.

In order to schedule these unallocated tasks, two approaches can be considered: portioned scheduling and scheduling at job boundaries (*i.e.* restricted migration). To illustrate the first one, we show in Figure 1 four tasks

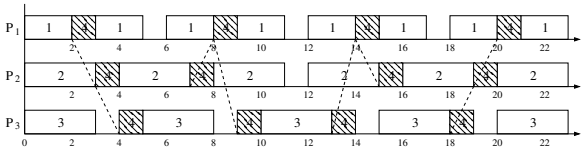

Figure 1. Portioned scheduling.

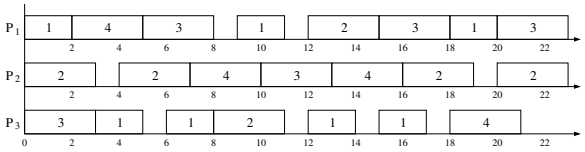

Figure 2. Scheduling at job boundaries.

$\{\tau_1(2,3), \tau_2(3,4), \tau_3(3,5), \tau_4(3,6)\}$ scheduled on three processors. No partition can be found because $u_i + u_j > 1,\ \forall i \neq j,\ i,j \in \{1,2,3,4\}$ ($u_i = \frac{WCET}{period}$). In this example, the tasks are allocated in decreasing utilization order and τ_4 is the last one. Therefore τ_4 is portioned and its jobs are scheduled on the three processors.

This approach can be difficult to implement because it is not easy to split a job in many parts. A job consists in pieces of code which can contain indivisible sections. Moreover, a large number of migrations occurs since a job can migrate several times. The second approach is attractive because the migrations are at job boundaries. Therefore no migration overhead are incurred if the consecutive jobs of a task don't share any data. We propose to plit a task in several k subtasks. Each subtask has the same WCET and deadline as its parent task but the subtasks are less frequently activated (their periods are multiplied). $k-1$ subtasks are unsynchronized with an offset to avoid the jobs overlaps. We represent in Figure 2 the same example as in Figure 1. For instance, the subtasks τ_1^1 (0, 2, 3, 9) on P_1, τ_1^2 (3, 2, 3, 9) on P_3 and τ_1^3 (6, 2, 3, 9) on P_3 produce the same execution as τ_3 (2, 3) (where (w, x, y, z) is (*offset*, *WCET*, *deadline*, *period*)).

Open problem. We consider the case where a partitioning algorithm doesn't succeed to partition the set τ of tasks. Our aim is to build an algorithm which can split the pending tasks in subtasks such that a feasible partition of τ can be found. If no pending task can be split, already allocated tasks can be split until a feasible partition is found. The problem is to build an algorithm which finds a valid splitting scheme if such a scheme exists such that a feasible partition can be found.

Insights. We want to build an algorithm which splits the tasks such that the migration occurs at job boundaries. A first approach is to build a static-priority restricted migration scheduler. This scheduler allocates jobs of the tasks when they are activated on a ready processor according to a static-priority scheme. By logging the allocations made by this scheduler, we can build our splitting scheme. Notice that, this approach can be complex because it may require the consideration of an interval of study which can be exponential. Moreover, it is not obvious to decide on which processor a task must be activated to minimize the number of subtasks.

References

[1] N. W. Fisher, S. K. Baruah, and T. P. Baker, "The partitioned scheduling of sporadic tasks according to static-priorities," in *Proceedings of the 18th Euromicro Conference on Real-time Systems (ECRTS)*. Dresden, Germany: IEEE Computer Society, July 2006, pp. 118–127.

[2] B. Andersson and J. Jonsson, "Preemptive multiprocessor scheduling anomalies," in *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS)*. Fort Lauderdale, Florida, USA: IEEE Computer Society, April 2002, pp. 12–19.

[3] R. I. Davis and A. Burns, "Robust priority assignment for fixed priority real-time systems," in *Proceedings of the 28th IEEE Real-Time Systems Symposium (RTSS)*. Tucson, Arizona, USA: IEEE Computer Society, December 2007, pp. 3–14.

[4] L. Bougueroua, L. George, and S. Midonnet, "Dealing with execution-overruns to improve the temporal robustness of real-time systems scheduled FP and EDF," in *Proceedings of the 2nd International Conference on Systems (ICONS)*. Sainte-Luce, Martinique: IEEE Computer Society, April 2007, p. 8pp.

[5] F. Fauberteau, S. Midonnet, and L. George, "A robust partitioned scheduling for real-time multiprocessor systems," in *Proceedings of the 7th IFIP TC 10 Working Conference on Distributed and Parallel Embedded Systems (DIPES)*. Brisbane, Australia: Springer Science and Business Media, September 2010, pp. 193–204.

[6] S. Kato and N. Yamasaki, "Portioned static-priority scheduling on multiprocessors," in *Proceedings of the 22th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. Miami, Florida, USA: IEEE Computer Society, April 2008, pp. 1–12.

[7] S. Kato and N. Yamasaki, "Semi-partitioned fixed-priority scheduling on multiprocessors," in *Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. San Francisco, California, USA: IEEE Computer Society, April 2009, pp. 23–32.

[8] S. K. Baruah and J. Carpenter, "Multiprocessor fixed-priority scheduling with restricted interprocessor migrations," in *Proceedings of the 15th Euromicro Conference on Real-time Systems (ECRTS)*. Porto, Portugal: IEEE Computer Society, July 2003, pp. 195–202.