Exploiting Uni-Processor Schedulabilty Analysis for Partitioned Task Allocation on Multi-Processors with Precedence Constraints

Mario Bambagini, Giorgio Buttazzo
{mario.bambagini, giorgio.buttazzo}@sssup.it
Scuola Superiore Sant'Anna
Pisa, Italy

Sverre Hendseth
sverre.hendseth@itk.ntnu.no
Norwegian University of Science and Technology
Trondheim, Norway

***Abstract*—This paper considers the problem of scheduling real-time tasks with precedence and communication constraints on heterogeneous multiprocessor systems. Most partitioned approaches statically schedule the task set by computing start times and finishing times for each task in such a way that a desired cost function is minimized. The resulting optimization problem is however highly complex. The open problem proposed in this paper is to reduce the overall complexity by transforming precedence relations into real-time constraints and exploit uniprocessor scheduling results to guarantee the task set.**

I. Introduction

The problem of task allocation and scheduling in multiprocessor systems under precedence and real-time constraints is known to be NP-Hard and has been investigated for many years.

Such a problem has become dominant with the development of Multi-Processors System-on-Chips (MPSoC), distributed embedded systems, and computer clusters. In spite of the different contexts, the common goal is to provide algorithms for the automatic allocation of tasks to optimize the computational resources.

Many algorithms [1] have been proposed in the literature and they can be divided into global and partitioned approaches. Global algorithms pick the highest priority task from a single ready queue (shared by the cores) and allocate it on an available processor. Partitioned approaches first allocate the task on the processors and then schedule them using a local scheduler. A wide range of algorithms exist, which spread from complete searches [2], [3], meta-heuristics [4], and heuristics [5].

Most of these approaches start from a task set with precedence and time constraints and produce a static schedule, stored in a table and executed in a time-triggered fashion. The approach considered in this paper proposes to transform precedence relations into activation times and deadlines for each tasks and use an online scheduling algorithm to execute them. The advantage of this approach is to exploit existing uniprocessor results for analyzing the schedulability of the task set allocated on each processor, thus reducing the complexity to find a feasible solution.

II. Model

We consider a set Φ of m heterogeneous processors and a set Γ of n preemptive real-time tasks, characterized by a set of precedence constraints.

Each processor ϕ_j has a specific type pt_j which collects processors in groups of performance. The processors are assumed to be linked together through a fully-connected network which consists of a dedicated communication link for each node pair. These links are assumed to be full-duplex and heterogeneous, meaning that data transfer may take different time for the same message size, depending on where tasks are allocated.

A task τ_i allocated on processor ϕ_j is characterized by a worst-case computation time $C_{i,j}$. Computation times of τ_i are equal for the same processor type. Precedence dependencies are represented by a direct acyclic graph. More precisely, the notation $\tau_i \rightarrow \tau_j$ indicates that τ_j has to start not earlier than τ_i finishing time plus the communication time required for data exchange. The data transmission delay is computed as the amount of data exchanged between τ_i and τ_j divided by the bandwidth guaranteed between the two hosting processors. The whole application is considered to be periodic with a period P and a relative deadline D. For the sake of simplicity, P and D are assumed to be equal. We assume that in each processor tasks are scheduled by Earliest Deadline First (EDF) [6].

III. Open Question

Most of the proposed partitioned algorithms produce a static allocation and then a static schedule on each processor, and the application feasibility is guaranteed if and only if the latest finishing time is less than or equal to the application deadline.

Our goal is to split the multi-processor scheduling problem into m different uni-processor scheduling problems and exploit the well-known theoretical results to guarantee the feasibility on each processor. In order to apply this approach, however, it is necessary to assign an activation time and a deadline to each task, so that EDF can schedule them.

For each processor, the feasibility can be checked using the Processor Demand Criterion [7] or through the offset analysis [8]. In this case the analysis is simplified because task periods

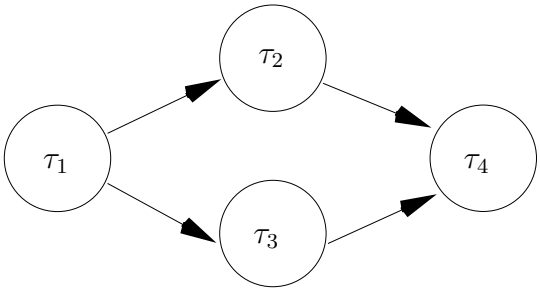

Fig. 1. Task Graph

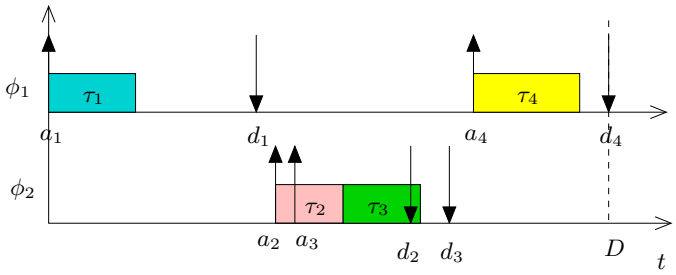

Fig. 2. A possible assignment

are all equal to the application period. This also means that the hyperperiod is equal to the application period and only one instance for each task must be taken into account within the analysis interval.

In this way, the problem mainly consists in an allocation phase and a time assignment phase, leaving the scheduling to EDF. While the allocation has been investigated for a long time, providing an effective assignment of activation times and deadlines is still an open question.

An example is reported in Figure 2 to show a possible timing constraints assignment from the precedence relations depicted in Figure 1. In the example, four tasks must be executed on two processors running EDF as a scheduler. τ_1 and τ_4 are allocated on the first processor ϕ_1 and τ_2 and τ_3 on ϕ_2. A valid timing assignment must set τ_1's deadline earlier than the activations of τ_1's successors (also considering the communication overheads). The same for τ_4.

A similar assignment problem has been addressed by Buttazzo et al. [9], who extended the idea proposed by Chetto et al. [10]. Their method, however, focuses more on the path analysis rather than on each single task and their solution assumes homogeneous systems and negligible communication costs. Moreover, for the task set illustrated in the example, assigning each flow to a different processor, their method would make τ_4's activation time occur $c_{4,1}$ units of time earlier than its deadline, meaning that τ_4 has a relative deadline equal to its computation time (Figure 3). Since their procedure would set $a_3 = a_2$ and $d_3 = d_2$, the task set would be infeasible on ϕ_2, leading τ_3 to miss its deadline. In fact, $d_3 - a_3 = d_2 - a_2 > c_{2,2} + c_{3,2}$, although a considerable amount of time is wasted till d_1.

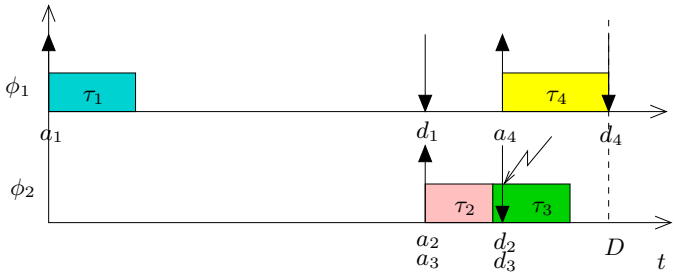

Fig. 3. Buttazzo et al.'s assignment

This example suggests that it is worth investigating alternative approaches to assign activation time and deadlines that guarantee feasibility while minimizing a desired cost function.

References

[1] R. I. Davis and A. Burns, "A survey of hard real-time scheduling for multiprocessor systems," *ACM Comput. Surv.*, vol. 43, no. 4, pp. 35:1–35:44, Oct. 2011. [Online]. Available: http://doi.acm.org/10.1145/1978802.1978814

[2] I. C. S. Institute, K. Shin, and D. Peng, *Static Allocation of Periodic Tasks with Precedence Constraints i Distributed Real-time Systems*, ser. Technical report (International Computer Science Institute). International Computer Science Institute, 1988. [Online]. Available: http://books.google.it/books?id=P3iIGwAACAAJ

[3] M. Lombardi and M. Milano, "Optimal methods for resource allocation and scheduling: a cross-disciplinary survey," *Constraints*, vol. 17, no. 1, pp. 51–85, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1007/s10601-011-9115-6

[4] V. V. Peteghem and M. Vanhoucke, "A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 201, no. 2, pp. 409–418, March 2010. [Online]. Available: http://ideas.repec.org/a/eee/ejores/v201y2010i2p409-418.html

[5] L.-C. Canon, E. Jeannot, R. Sakelariou, and W. Zheng, "Comparative Evaluation of the Robustness of DAG Scheduling Heuristics," in *Integration Research in Grid Computing, CoreGRID integration workshop*, S. Gorlatch, P. Fragopoulo, and T. Priol, Eds. Hersonissos, Crete, Grèce: Crete University Press / Springer US, 2008, pp. 63–74. [Online]. Available: http://hal.inria.fr/inria-00333904

[6] C. L. Liu and J. W. Layland, "Readings in hardware/software co-design," G. De Micheli, R. Ernst, and W. Wolf, Eds. Norwell, MA, USA: Kluwer Academic Publishers, 2002, ch. Scheduling algorithms for multiprogramming in a hard-real-time environment, pp. 179–194. [Online]. Available: http://dl.acm.org/citation.cfm?id=567003.567018

[7] S. K. Baruah, R. R. Howell, and L. Rosier, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," *Real-Time Systems*, vol. 2, pp. 301–324, 1990.

[8] R. Pellizzoni and G. Lipari, "Feasibility analysis of real-time periodic tasks with offsets," *Real-Time Systems*, vol. 30, no. 1-2, pp. 105–128, 2005.

[9] G. C. Buttazzo, E. Bini, and Y. Wu, "Partitioning real-time applications over multicore reservations," *IEEE Trans. Industrial Informatics*, vol. 7, no. 2, pp. 302–315, 2011.

[10] H. Chetto, M. Silly, and T. Bouchentouf, "Dynamic scheduling of real-time tasks under precedence constraints," *Real-Time Syst.*, vol. 2, no. 3, pp. 181–194, Sep. 1990. [Online]. Available: http://dx.doi.org/10.1007/BF00365326