The Journal of Systems and Software 119 (2016) 70–86

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Real-time hierarchical systems with arbitrary scheduling at global level

Ana Guasque, Patricia Balbastre[1,*], Alfons Crespo

Universitat Politècnica de València, Valencia, ES, 46022, Spain

ARTICLE INFO

Article history:
Received 22 February 2016
Revised 4 May 2016
Accepted 26 May 2016
Available online 27 May 2016

Keywords:
Real-time systems
Partitioned systems
Embedded systems
Real-time systems scheduling
Real-time algorithms

ABSTRACT

Partitioned architectures isolate software components into independent partitions whose execution will not interfere with other partitions, preserving temporal and spatial isolation. Hierarchical scheduling can effectively be used to schedule these systems. Schedulability analysis of hierarchical real-time systems is based on prior knowledge of the local and the global scheduling algorithms.

In a partitioned system with safety and security issues and certification assurance levels, global scheduling is usually generated using a static table. Therefore, each partition must allocate task jobs only in the temporal windows reserved for that partition. Even if the static table can come originally from a periodic server or other scheduling policy, the final plan may be modified due to changes in the system requirements. As a consequence, the CPU assignment to a partition does not have to correspond to any known policy. In this case, it is not possible to use existing scheduling analysis for hierarchical systems.

This paper studies a new scheduling problem: a hierarchical system in which global policy is not known but provided as a set of arbitrary time windows.

1. Introduction

In many domains such as avionics, space or industrial control systems, hard real-time constraints, safety and security issues and certification assurance levels are commonly required. Integrated Modular Avionics (IMA) is an architectural proposal that emerged as a design concept to integrate several applications with different levels of criticality in a hardware platform. The IMA approach proposes to encapsulate functions into partitions configuring a partitioned system. Partitioned architectures isolate software components into independent partitions whose execution must not interfere with others, preserving temporal and spatial isolation. Several projects have been successfully developed using this approach in the avionic market.

In the last decade, the European space sector has adapted the initial IMA approach for the space requirements for the new generation of satellites (Windsor and Hjortnaes, 2009). The IMA-SP project focused on mono-processors (IMA-SP, 2011-13a). The platform defines a virtualization layer (hypervisor) that permits execution of several partitions. Each partition can contain a guest operating system and the application software. The hypervisor is in charge of ensuring temporal and spatial isolation of partitions.

An IMA development process involves several roles like:

- System Architect (SA): The SA is responsible for defining the overall system requirements and system design, including optimal decomposition into hosted partitions jointly with the detailed resource allocation per partition.
- System Integrator (SI): The SI is responsible for verifying the feasibility of the system requirements defined by the SA, as well as responsible for the configuration and integration of all components.
- Application Suppliers (AS): An AS is responsible for developing an application according to the overall requirements from the SA and the SI. AS must verify compliance with the allocated budget and safety parameters. Assuming that each application is located in a partition and a partition can have only one application, an AS can also be called Partition Developer (PD).

There are other roles in the process but due to space restrictions we only detail those interesting for the purpose of this article. For a complete description of the main roles and responsibilities see (IMA-SP, 2011-13b).

* Corresponding author.
E-mail addresses: anguaor@ai2.upv.es (A. Guasque), patricia@ai2.upv.es (P. Balbastre), acrespo@ai2.upv.es (A. Crespo).

[1] This work has been funded by the Spanish government under grant TIN2014-56158-C4-1-P-AR and by the European Commission under FP7-ICT-2013.3.4 Programme with grant 610640

http://dx.doi.org/10.1016/j.jss.2016.05.040
0164-1212/

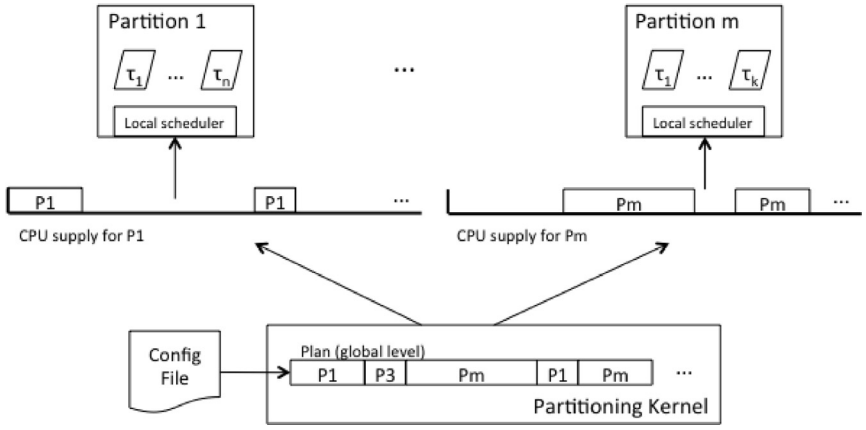

Fig. 1. General overview of the partitioned system.

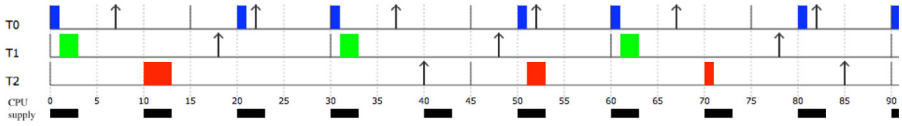

Fig. 2. Execution chronogram and CPU supply of a partition.

A key element in the development process and the final execution is the configuration of the system defined by the SA, which includes the description of the components and resource allocation. This is identified as configuration data or configuration file. In order to preserve the confidentiality of the development process, configuration data is split and delivered to each PD with the required information for developing the application.

The SI is responsible for CPU allocation of temporal resources to applications while the PD manages the time budget assigned to its tasks by the SI. Based on the proposed software architecture in an IMA system where a hypervisor supports the execution of several temporal and spatial isolated partitions, the system can be modeled as a hierarchical real-time system in which tasks are allocated to partitions. The SI allocates CPU in the global level, according to the scheduling algorithm of its choice, while the PD internally schedules tasks with its own scheduling algorithm and the assigned CPU budget. The SI is responsible for ensuring feasibility in the global level while PD ensures feasibility in the corresponding local level. Fig. 1 shows the structure of a partitioned system. The scheduling plan in the global level schedules partitions according to an offline plan defined in the static configuration file of the system.

Thus, a partition does not have all the time assigned to schedule its tasks, but only certain slots throughout the hyper-period. An example is shown in Fig. 2, where a partition with a set of periodic tasks is scheduled under EDF (Earliest Deadline First) policy.

In the previous figure, the global scheduler peridically assigns the CPU to the task set, that is, the partition receives a periodic resource supply that provides 3 units of CPU every 10 units of time. The black rectangles at the bottom of the figure represent the slots assigned to the partition. Obviously, tasks cannot execute outside these slots, since they are reserved for other partitions.

The list of assigned slots is provided by the SI, responsible for ensuring the feasibility in the global level. Thus, PD gives the SI its temporal requirements, normally in the form of CPU bandwidth. The SI calculates and assigns this bandwidth to partitions using a well known bandwidth server or cyclic scheduling. ARINC 653 standard (Avionics Application Software Standard Interface, ARINC-653) defines a hierarchical scheduling where a static cyclic executive scheduler is used in the global level.

If the assignment is made using a bandwidth algorithm or a periodic resource model, the corresponding feasibility tests are available in the literature so the PD can apply them to know if its tasks are schedulable with this slots assignment (see Section 8). On the contrary, if the SI makes the asignment arbitrarily (i.e. not following any existing scheduling algorithm) the authors are not aware of any article that addresses and solves this problem.

Below, we present an example of why a partition can be assigned an arbitrary sequence of slots. Let us assume a partitioned system with three partitions (P1, P2 and P3) and the scheduling plan shown in Fig. 3(a).

If the temporal requirements of P2 changefor any reason, P2 will be scheduled in the empty slots not used by P1 and P3 (Fig. 3(b)). These slots do not correspond to any periodic reservation so we can consider that the sequence of slots provided by SI to PD of P2 are arbitrary. Of course, we can also re-schedule the entire system but then the scheduling of P1 and P2 would change, requiring certification of partitions whose requirements do not change. Such an effort must be avoided if possible.

If we later add a fourth partition to the system (P4) we will have to schedule P4 in the idle slots not used by P1, P2 and P3. Again, the slots reserved for P4 can be considered arbitrary (Fig. 4(b)).

These two situations show two different scenarios where a partition must be scheduled in time slots that do not follow any known allocation. Schedulability tests for hierarchical systems are based on calculating the worst case response time of tasks in the local level and adding the worst case overhead due to the global level. This last overhead cannot be calculated if the scheduling policy in the global level is not known. Thus, the existing literature does not give a solution to this problem. For this reason, we provide a solution to analyze the schedulability of a task set of a partition where the scheduling algorithm is arbitrary at global level.

1.1. Contributions and outline

The problem to be addressed is concerned with the schedulability of a hierarchical system composed of two levels. The global level policy is not known but provided by the SI as a set of arbitrary time slots. By arbitrary we understand that the sequence of