Schedulability Analysis in Partitioned Systems for Aerospace Avionics*

João Craveiro and José Rufino
University of Lisbon, Faculty of Sciences, LaSIGE
Campo Grande, 1749-016 Lisboa, Portugal
jcraveiro@lasige.di.fc.ul.pt, ruf@di.fc.ul.pt

Abstract

Aerospace mission systems' size, weight and power consumption requirements call for the integration of multiple functions on a single embedded computing platform. A current trend to guard against potential timeliness and safety issues in integrating applications of different natures and providers is the employment of temporal and spatial partitioning. The AIR architecture, defined within initiatives sponsored by the European Space Agency to meet these goals, supports multiple partition operating systems, and advanced timeliness control and adaptation mechanisms. In this paper we propose how to take advantage of composability properties inherent to the build and integration process of AIR-based systems, towards tool-assisted scheduling analysis and configuration support.

1. Introduction

Size, weight and power consumption requirements of systems aboard space missions call for the integration of multiple functions on a single embedded computing platform. In order to minimize robustness drawbacks of this integration, onboard applications are functionally separated in logical containers, called partitions. With partitioning we achieve two results; faults can be contained to the domain in which they occur, and independent verification and validation of software components becomes possible, easing the overall certification process, fundamental for aerospace avionics. The safety of this approach is enforced through robust temporal and spatial partitioning (TSP). Temporal partitioning concerns partitions not interfering with the fulfilment of each other's real-time requisites, while spatial partitioning encompasses the usage of separate memory addressing spaces dedicated to each partition. Prompted by the strong interest from the space agencies in applying TSP to the aerospace domain, the *AIR (ARINC 653 In Space Real-Time Operating System) architecture* was defined [8].

This paper discusses our ideas of introducing independent validation of scheduling requirements and tool-assisted generation of partition scheduling tables into the AIR build and integration process. This is allowed by AIR's composability properties. Section 2 provides background on the AIR technology [8]. Section 3 addresses the main contribution of this paper, scheduling analysis in TSP systems. Related work is exposed in Section 4. Section 5 concludes with final remarks and future work.

2. The AIR Architecture

The research leading up to the current state of the art has been motivated by the interest sparked by several space industry partners [9, 12].

2.1. System architecture

The AIR architecture (Figure 1) has been designed to fulfil the requirements for robust TSP, and foresees the use of different operating systems among the partitions, either real-time operating systems (RTOS) or generic non-real-time ones; each of these, regardless of their nature, will be herein referred to as *partition operating system* (POS).

The *AIR Partition Management Kernel (PMK)* ensures robust temporal and spatial partitioning [8].

The *Application Executive (APEX)* provides a standard interface between applications and the underlying core software layer (Figure 1), in conformity with the ARINC 653 specification [1]. The advanced notion of

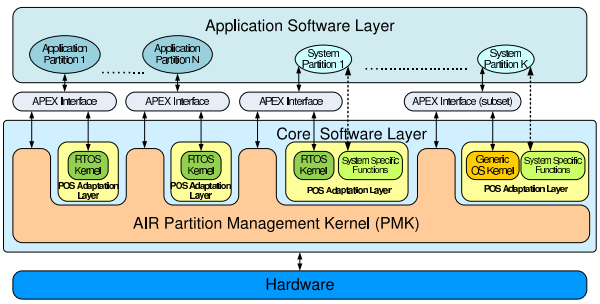

Figure 1. AIR system architecture

*This work was partially developed within the scope of the European Space Agency Innovation Triangle Initiative program, through ESTEC Contract 21217/07/NL/CB, Project AIR-II (ARINC 653 in Space RTOS — Industrial Initiative, http://air.di.fc.ul.pt). This work was partially supported by Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology), through the Multiannual Funding and CMU-Portugal Programs and the Individual Doctoral Grant SFRH/BD/60193/2009.

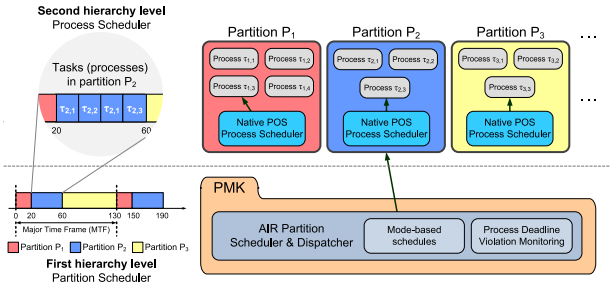

Figure 2. AIR two-level hierarchical scheduling

Portable APEX [10] exploits availability of *AIR POS Adaptation Layer (PAL)* functions. The AIR PAL wraps each POS, providing a common interface to the surrounding components. Changes needed to support a new family or version of a POS are circumscribed, and previous or ongoing verification, validation and/or certification efforts are not hindered. Interpartition communication services are provided by APEX and supported by AIR PMK [8].

2.2. Temporal partitioning

Temporal partitioning is achieved through a two-level hierarchical scheduling scheme (Figure 2). In the first level, partitions are scheduled over a cyclic sequence of fixed time slices. Inside each partition, processes compete according to the POS kernel's native process scheduler.

AIR implements the advanced notion of *mode-based partition schedules*, allowing temporal requirements to vary according to the mission's phase or mode of operation. The system has a set of partition schedules among which it can switch during its operation [8].

2.3. System model

We present here a simplified version of the AIR system model defined in [8]. Its notation is summarized in Table 1. The system comprises a set of partitions $P = \{P_1, \ldots, P_{n(P)}\}$, where:

$$P_m = \langle \tau_m \rangle \tag{1}$$

characterizes a partition by its taskset (processes) τ_m. Given the support for mode-based schedules, the partitions' timing requirements are not a characteristic of the partition per se, but rather depend on each schedule.

The integrator-defined set $\chi = \{\chi_1, \ldots, \chi_{n(\chi)}\}$ contains the partition scheduling tables (PST) among which the system can switch. Each schedule:

$$\chi_i = \langle MTF_i, Q_i, \omega_i \rangle \tag{2}$$

is cyclically repeated over its major time frame duration MTF_i. Each element in Q_i:

$$Q_{i,m} = \langle P^{\chi}_{i,m}, \eta_{i,m}, d_{i,m} \rangle \tag{3}$$

Table 1. Notation used

Symbol	Description
	Convention used
$\mathbb{N}$	Set of natural numbers ($0 \notin \mathbb{N}$)
$n(S)$	(Where S is a set) Equivalent to $\#S$.
	Partitions
P	Set of partitions in the system
P_m	Partition m
τ_m	Taskset (processes) of partition P_m
	Partition scheduling
χ	Set of partition schedules available
χ_i	Partition schedule i
MTF_i	Major time frame of schedule χ_i
Q_i	Set of partition time requirements for χ_i
$P^{\chi}_{i,m}$	Each partition with at least one time window in χ_i
$\eta_{i,m}$	Activation cycle of partition $P^{\chi}_{i,m}$ under χ_i
$d_{i,m}$	Duration of partition $P^{\chi}_{i,m}$ under χ_i
ω_i	Set of time windows in schedule χ_i
$\omega_{i,j}$	Time window j in schedule χ_i
$P^{\omega}_{i,j}$	Partition active during time window $\omega_{i,j}$
$O_{i,j}$	Offset of $\omega_{i,j}$, relative to the beginning of MTF_i
$c_{i,j}$	Duration of $\omega_{i,j}$

expresses timing requirements for each partition $P^{\chi}_{i,m} \in P$ which is scheduled under χ_i: its cycle $\eta_{i,m}$ and duration (required time per cycle) $d_{i,m}$. ω_i is a set of time windows:

$$\omega_{i,j} = \langle P^{\omega}_{i,j}, O_{i,j}, c_{i,j} \rangle \qquad P^{\omega}_{i,j} \in Q_i \tag{4}$$

characterized by their offset $O_{i,j}$ and duration $c_{i,j}$. We assume windows, while not being necessarily contiguous, do not intersect:

$$O_{i,j} + c_{i,j} \leq O_{i,j+1} \qquad \forall j < n(\omega_i) \tag{5}$$

and are fully contained inside the duration of one MTF:

$$O_{i,n(\omega_i)} + c_{i,n(\omega_i)} \leq MTF_i \ . \tag{6}$$

We also assume MTF_i is a multiple of the cycle duration of every partition with at least one window in χ_i:

$$MTF_i = k_i \times \operatorname*{lcm}_{Q_{i,m} \in Q_i} (\eta_{i,m}) \qquad k_i \in \mathbb{N} \ . \tag{7}$$

3. Schedulability analysis in TSP systems

3.1. Mutual impact within the two-level scheduling

The first step in this work pertains to the analysis of the mutual impact between the two levels of the hierarchical scheduling scheme of a TSP system (Figure 2). The expected outcomes include theoretical results and tools which take these results into consideration, in order to help system integrators analyse and define partition scheduling tables.

On the one hand, we have to analyse the restrictions that each partition — processes' characteristics, process scheduling algorithm, or input/output (I/O) and interpartition communication behaviour — impose on the definition of the partition scheduling tables. For a schedule

$\chi_i = \langle MTF_i, Q_i, \omega_i \rangle$ to be feasible considering the timing requirements of the partitions it includes, the following condition must hold:

$$\sum_{\{\ \omega_{i,j} \in \omega_i | P^{\omega}_{i,j} = P_m \wedge O_{i,j} \in [k\,\eta_m ; (k+1)\eta_m [\ \}} c_{i,j} \geq d_m \qquad \forall P_m \in Q_i\ , \forall k \in \left[0..\frac{MTF_i}{\eta_m} - 1\right] \ . \tag{8}$$

It is however necessary that timing requirements for each partition P_m under a given schedule match the characteristics of the respective processes τ_m. These range from process parameters like period or minimum interarrival interval, deadline, and worst case execution time (WCET) to additional requisites such as I/O and interpartition communication blocking time. Our goal is to devise deterministic ways to automate the transformation of process characteristics into partition timing requirements and these into PSTs. This last step should take into consideration both temporal and functional dependencies.

On the other hand, there is a fundamental need to study the impact of the cyclic partition scheduling on the timeliness, performance, and overall functioning of applications running on partitions. While this analysis is of paramount importance for real-time applications, non-real-time applications also benefit vastly from it. Despite not having hard deadlines to fulfil, the latter must have a minimum performance threshold for their inclusion to be beneficial and useful.

3.2. Composability

The modularity of the AIR architecture design and of its build and integration process enables *composability* of AIR-based systems. The several components that may compose such a system can be developed, verified and validated independently. This eases certification efforts, since only modified modules need to be reevaluated.

From the point of view of one partition's provider, this further signifies that development and validation does not depend on knowledge of the other partitions (individually or as a whole). At most, the development of one partition should be aided by a set of guidelines for its applicability to the target TSP systems in general.

The system integrator is responsible for guaranteeing a correct partition scheduling, so that partitions and the system as a whole meet their timing requisites.

3.3. Build and integration process

Schedulability results for TSP systems allow extensions to the software build and integration processes of AIR-based systems, for the benefit of both application developers and system integrators.

Application developers may have a scheduling analysis phase introduced in their production cycle (Figure 3a). The goal is for developers to be able to analyse the feasibility of their applications provided the timing requirements (period, WCET, deadline, etc.) of their processes.

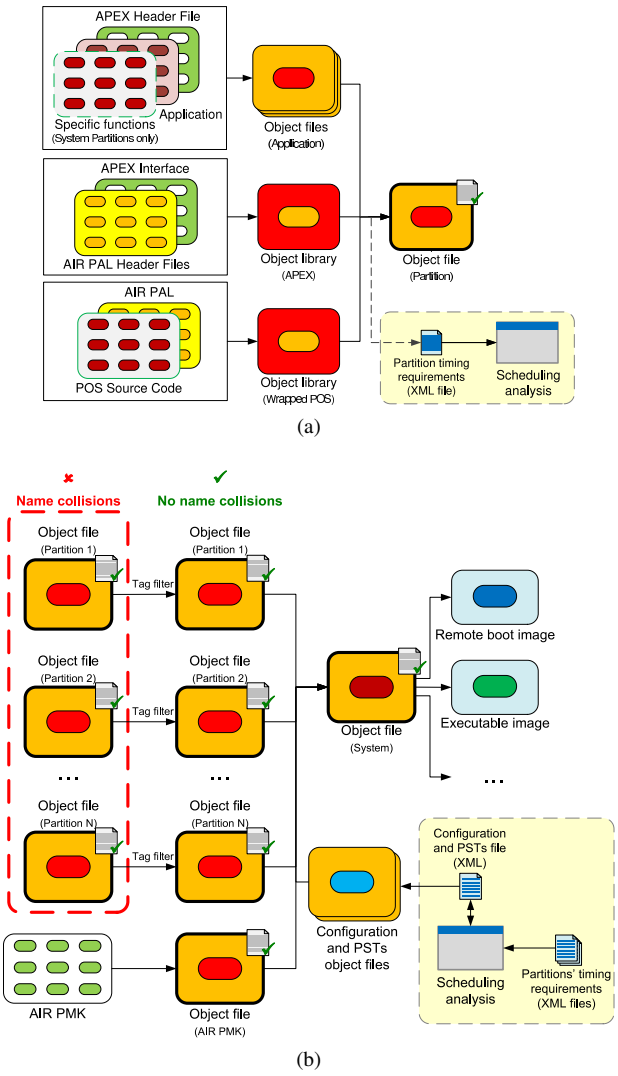

Figure 3. Introduction of scheduling analysis features for (a) application developers, and (b) system integrators

This information can be either estimated by the developers or tentatively determined through static code analysis [7].

In the system integration phase, scheduling analysis capabilities shall be introduced in relation with the generation of a system-wide configuration (Figure 3b) [1].

The tool allows the system integrator to load a previously created XML (Extensible Markup Language) configuration file, and simulate or analyse for feasibility the PSTs in χ. In case of modifications, the XML configuration file could be exported (keeping configurations other than partition scheduling unchanged) and used to generate the configuration object files. Support for more advanced capabilities would allow the system integrator to load both an XML configuration file (with incomplete or absent information regarding PSTs), and a set of XML files with the partitions' timing requirements (Figure 3), such that:

$$\mathsf{TimingRequirements}_m \supseteq \tau_m \qquad \forall m \leq n(P) \ . \tag{9}$$

Since the XML configuration file may not contain all or any of the PSTs which will feature in the system, the relation between its contents and the set χ of PSTs can be formally expressed as:

$$\mathsf{ConfigurationFile} \cap \chi \subseteq \chi \ . \qquad (10)$$

In the extreme case, the XML configuration file initialy provided contains no PSTs at all and these will be created with the aid of the scheduling tool ($\mathsf{ConfigurationFile} \cap \chi = \emptyset$). From these timing requirements, the tool would assist the system integrator with the generation of one or more PSTs fulfilling the partitions' requirements. These PSTs would then be combined with the remaining configuration parameters and be exported to a complete XML configuration file, from which in turn the configuration and PSTs object files would be derived (Figure 3b).

3.4. Conception of a scheduling tool

The introduction of TSP-specific scheduling analysis capabilities can be provided through a tool developed from scratch, or by extending an existing tool. We are looking into the possibility of extending Cheddar [11] to this purpose. Cheddar provides a set of scheduling algorithms and policies on which it is capable of performing feasibility tests and/or scheduling simulations.

Cheddar permits TSP schedulability analysis, but in a limited and non-user-friendly way. The definition of PSTs is done by modifying the source code of the simulated partition scheduler [11].

The enhancements we envision for Cheddar include adding support for importing and exporting platform configuration and PSTs files (e. g., XML), assisted by a graphical user interface for the system integrator to easily and flexibly modify PSTs.

4. Related work

Real-time scheduling in general is a widely studied subject, with research dating back to the early 1970s [6]. However, scheduling analysis for TSP systems is a more recent subject and thus fewer literature can be found.

Some design alternatives proposed in [2] are not compatible with the principles and goals of modern TSP systems, such as assigning processes to partitions according to their periods (instead of their functionality and/or degree of criticality). The execution of applications in a partitioned environment can be equated to the execution on a slow processor [3, 5]. However, these approaches do not fulfil the requirements of emerging TSP systems.

The timing analysis in [4] relies on a model with some limiting (and, in some cases, unjustified) assumptions; for instance, the authors ignore aperiodic processes on the grounds that they are scheduled as background workload.

Some of these works [2, 5] build upon a simplified model of a TSP system, by assuming each partition has one execution time window per each one of its periods.

5. Conclusion and future work

We discussed rules, techniques and tools to introduce schedulability analysis and tool-assisted generation of partition scheduling tables into the build and integration process of AIR-based systems. This takes profit from the composability properties allowed by the AIR architecture design and the build and integration process itself.

Future work includes the extension of these concepts to allow multicore-enabled TSP systems, which feasibility shall be backed up by both schedulability and safety considerations. The hereby proposed tools should then accommodate multicore support as yet another scheduling analysis parameter.

References

[1] AEEC. Avionics application software standard interface, part 1 - required services. ARINC Specification 653P1-2, Mar. 2006.

[2] N. Audsley and A. Wellings. Analysing APEX applications. In *Proc. 17th IEEE Real-Time Systems Symp.*, pages 39–44, Washington, DC, USA, Dec. 1996.

[3] Z. Deng and J. W.-S. Liu. Scheduling real-time applications in an open environment. In *Proc. 18th IEEE Real-Time Systems Symp.*, pages 308–319, Dec. 1997.

[4] A. Easwaran, I. Lee, O. Sokolsky, and S. Vestal. A compositional scheduling framework for digital avionics systems. In *Proc. 15th IEEE Int. Conf. Embedded and Real-Time Computing Systems and Applications*, Beijing, China, Aug. 2009.

[5] Y. Lee, D. Kim, M. Younis, and J. Zhou. Partition scheduling in APEX runtime environment for embedded avionics software. In *Proc. 5th Int. Conf. Real-Time Computing Systems and Applications*, pages 103–109, Hiroshima, Japan, 1998.

[6] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[7] P. Pushner and C. Koza. Calculating the maximum execution time of real-time programs. *Journal of Real-Time Systems*, 1:160–176, Sept. 1989.

[8] J. Rufino, J. Craveiro, and P. Verissimo. Architecting robustness and timeliness in a new generation of aerospace systems. In A. Casimiro, R. de Lemos, and C. Gacek, editors, *Architecting Dependable Systems 7*, LNCS. Springer, 2010. Accepted for publication.

[9] J. Rushby. Partitioning in avionics architectures: Requirements, mechanisms and assurance. NASA Contractor Report CR-1999-209347, SRI International, California, USA, June 1999.

[10] S. Santos, J. Rufino, T. Schoofs, C. Tatibana, and J. Windsor. A portable ARINC 653 standard interface. In *Proc. IEEE/AIAA 27th Digital Avionics Systems Conf.*, St. Paul, MN, USA, Oct. 2008.

[11] F. Singhoff, J. Legrand, L. Nana, and L. Marcé. Cheddar: a flexible real time scheduling framework. *Ada Lett.*, XXIV(4):1–8, 2004.

[12] J.-L. Terraillon and K. Hjortnaes. Technical note on onboard software. European Space Technology Harmonisation, Technical Dossier on Mapping, TOSE-2-DOS-1, ESA, Feb. 2003.