Architecture, Mechanisms and Scheduling Analysis Tool for Multicore Time- and Space-Partitioned Systems

João Craveiro and José Rufino
Universidade de Lisboa, FCUL, LaSIGE
Lisbon, Portugal
Email: jcraveiro@lasige.di.fc.ul.pt, ruf@di.fc.ul.pt

Frank Singhoff
LISyC/University of Brest/UEB
Brest, France
Email: singhoff@univ-brest.fr

***Abstract*—Time- and space-partitioned systems (TSP) are a current trend in aerospace. They are employed to integrate a heterogeneous set of functions (different criticalities, real-time requirements, and origins) in a shared computing platform, fulfilling individual partitions' and global real-time properties. Applications are separated into logical partitions, scheduled according to predefined partition scheduling tables (PSTs). In this paper we expose our current work on exploiting multiprocessor/multicore processor platforms to add capacity, flexibility and safety to the current state of the art in TSP systems. We propose architectural evolutions, as well as the development of a schedule analysis and generation tool based on Cheddar. The tool will incorporate and extend real-time scheduling theory results, and be able to analyse the feasibility of PSTs and aid the generation of PSTs from the individual timing requirements of each function.**

***Keywords*-adaptive systems; aerospace industry; logic partitioning; processor scheduling; real time systems**

I. Introduction

The computing infrastructures supporting onboard aerospace systems, given the criticality of the mission being pursued, have strict dependability and real-time requisites. They also require flexible resource reallocation, and reduced size, weight and power consumption (SWaP). To cater to resource allocation and SWaP requirements, there has been a trend in the aerospace industry towards integrating the multiple hosted functions in the same computing platform. As these functions may have different degrees of criticality and predictability, and originate from multiple providers or development teams, safety issues might arise, which are mitigated by employing *time and space partitioning* (TSP). In TSP, onboard applications are functionally separated into logical containers — *partitions*. Partitioning allows containing faults in the domain in which they occur, and enables independent software verification and validation (easing the overall certification process). The issues of this paper are more tightly bound to the aspect of temporal partitioning, which ensures applications executing in one partition will not disrupt the use of any shared resource (most notably the processor) by applications in other partitions. This is essential to ensure the fulfilment of real-time guarantees and enable independent temporal analysis of the applications [1].

TSP concepts have been deployed in the civil aviation world, through the Integrated Modular Avionics (IMA) [2] and ARINC 653 [3] specifications. The interest from space industry partners in applying TSP concepts [4] originated the international consortium, sponsored by the European Space Agency (ESA), within which we took part in the development of the *AIR (ARINC 653 In Space RTOS) architecture* [5] — Section II.

In this paper we expose our current research regarding adding capacity, flexibility and safety to the current state of the art in TSP systems through the exploitation of platform equipped with either multiple processors or multicore processors. After a brief motivation based on the current state of the art (Section III), we propose extensions to the TSP principles using the AIR architecture as a reference — Section IV. Such added potential emphasizes the need for a scheduling tool to aid both application developers and system integrators, which we aim to cater to by extending Cheddar [6] — Section V.

II. AIR architecture for TSP systems in space

AIR [5] is designed to fulfil the requirements for robust TSP, and foresees the use of different partition operating systems (POS), either real-time or generic non-real-time ones. The modular design of the AIR architecture is shown in Figure 1. The *AIR Partition Management Kernel* (PMK) ensures robust temporal and spatial partitioning.

Temporal partitioning is achieved through a two-level hierarchical scheduling scheme shown in Figure 2. In the first level, partitions are scheduled cyclically over a *Major Time Frame* (MTF), according to a *partition scheduling table* (PST). In each partition, processes compete according

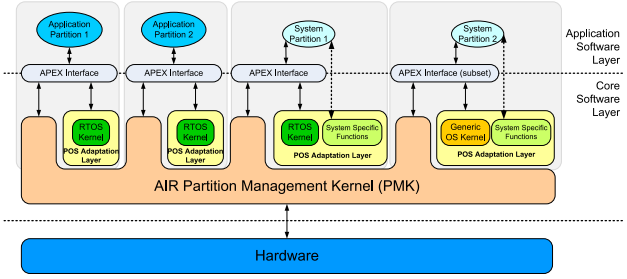

Figure 1. AIR architecture for TSP systems

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 FCT (through the Multiannual Funding and CMU-Portugal Programs, and the Individual Doctoral Grant SFRH/BD/60193/2009).

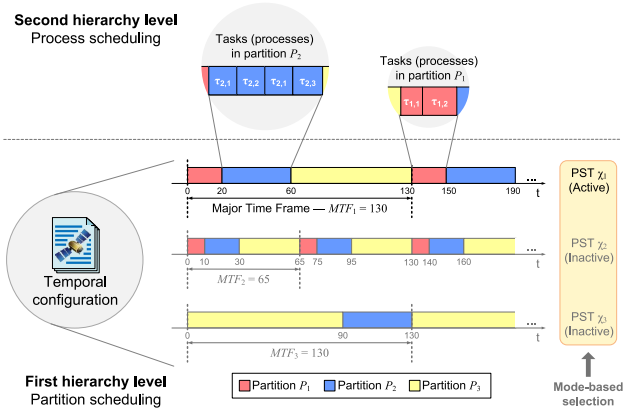

Figure 2. AIR two-level hierarchical scheduling, with support for mode-based schedules

to the native process scheduler of each POS. AIR supports mode-based partition schedules, among which the system can switch throughout its execution for (self-)adaptation to mission changes [7]. The *Application Executive* (APEX) provides a standard interface between applications and the underlying core software layer [3], [5].

III. Motivation and Related Work

Currently, neither commercial/proprietary TSP solutions nor academic research on TSP systems and architectures take profit from multicore platform to parallelly schedule multiple partitions [9].

Real-time scheduling theory literature on hierarchical scheduling with multiprocessor or multicore is scarce and recent, especially when compared with, on the one hand, multiprocessor real-time scheduling theory [8] and, on the other hand, the existent literature on uniprocessor hierarchical scheduling [9]. Researchers have proposed theoretical resource models for abstraction and composition of the supply–demand relationship between two levels of scheduling: the multiprocessor periodic resource (MPR) model [10], the multi supply function (MSF) and the Multi-(α, Δ) abstraction [11], the parallel supply function (PSF) [12], and the Bounded-Delay Multipartition (BDM) [13]. All of these require improvements to reach a state of full applicability to real industry-grade TSP scenarios.

Mollison et al. [14] illustrate the importance of supporting mixed-criticality workloads on multicore platforms in the context of avionics, but the encapsulation principles and hierarchical scheduling scheme are different from those used in TSP.

ARINC 653 [3] mentions possibility of multiple processors, but assumes a notion of "processor" which includes software components. This notion would imply replicating the core software layer — namely the PMK, in the specific case of the AIR architecture. The replicated instances would be independently integrated and configured, and the partitions associated with each one would be bound to a strong affinity to a processor core right from integration time. This approach limits the extent to which we can

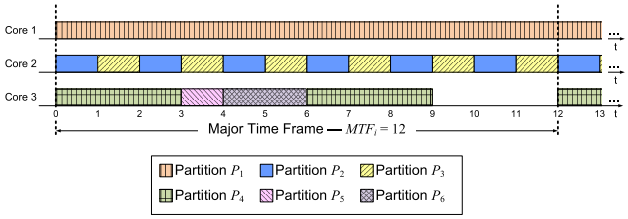

Figure 3. Interpartition parallelism example timeline

explore the (self-)adaptability mechanisms present in the AIR architecture, such as using redundancy and mode-based schedules to overcome a hardware failure [7], [15], or reconfiguring and/or updating the system in execution time [16].

IV. Multicore-aware AIR architecture for TSP systems

To add both the desired capacity and flexibility, we propose an architectural evolution consisting of one single instance of the AIR PMK, enhanced to take advantage of an underlying multiprocessor or multicore processor architecture. The association between partitions and cores is this more volatile than in an ARINC 653-like approach, as it can be expressed in configuration parameters designed to change dynamically (e. g., through mode-based partition schedules).

The AIR PMK may take advantage of a multiprocessor or multicore platform in several ways, either in alternative or cumulatively: *(i)* interpartition parallelism; *(ii)* intra-partition parallelism; *(iii)* enhanced spatial segregation; *(iv)* fault tolerance. We will now analyse the interpartition and intrapartition parallelism aspects in detail, as well as how they can be combined. Enhanced spatial segregation and fault tolerance mechanisms are out of the scope of this paper [15]. Furthermore, for now, we are not analysing multicore-specific issues in detail, such as the temporal effects of shared caches and bus contention. Therefore, both the multicore and the multiprocessor scenarios are addressed under the same abstraction.

A. Interpartition parallelism

Interpartition parallelism is achieved by extending the first level of the hierarchical scheduling scheme (Figure 2). At every given moment, more than one partition can be simultaneously active (scheduling and dispatching its processes), as long as those partitions do so on different processor cores. Figure 3 illustrates such a scenario. At every instant, there are always multiple cores in use by a different partition. Due to its timing characteristics, partition P_1 permanently takes over core 1.

Using, at first, an approach based on the partitioned multiprocessor real-time scheduling paradigm [17], we are looking into applying and extending previous research regarding feasibility tests [18]–[20], heuristics [13], [21], task sorting criteria, and the choice of these three parameters [22]. We also intend to weigh advantages and results with approaches based on the global multiprocessor real-time scheduling paradigm [17].

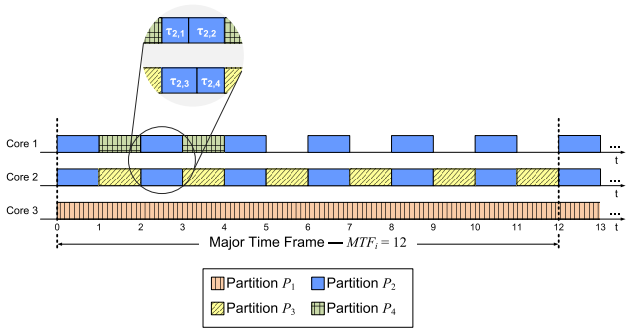

Figure 4. Example timeline with a combination of both inter- and intrapartition parallelism

B. *Intrapartition parallelism*

Another point of view under which we defend we can take profit from multicore platforms is allowing some partitions to simultaneously use more than one processor core during their active time windows. As such, a partition may parallelly execute multiple processes.

Since a partition will seldom take full advantage of intrapartition parallelism, the reasonable approach is to combine both interpartition and intrapartitition parallelism. By doing so, partitions which does not use all the processor cores made available open room for the execution of processes from other partitions (interpartition parallelism). Figure 4 illustrates this with an example timeline. Partition P_2 is the only partition to use more than one processor core. This way, it is able to run two processes simultaneously, as can be seen in the callout in Figure 4. Since P_2 does not use all the 3 cores, it also accommodates the parallel execution of the processes pertaining to partition P_1, which takes over core 3.

V. Schedule Analysis and Generation Tool for TSP System Integrators

The multiple applications that may compose such a TSP system can be developed, verified and validated independently. This eases certification efforts, since only modified modules need to be reevaluated. However, the developers of each component have to verify it taking into account that it will be integrated with other components of which they have no specific knowledge. At most, the development of one partition is only guaranteed to be aided by a set of guidelines for its applicability to the target TSP system (e.g. maximum percentage of time allowed to own computing resources). Developers of one application should be able to analyse its feasibility provided those guidelines and the timing requirements (period, worst-case execution time (WCET), deadline, etc.) of its tasks [23].

Further in the development process, the applications are integrated into the system. The way such an integration is performed with AIR is illustrated in Figure 5. A tag filter applied to the input from the partition application developers ensures there are no name collisions (e.g., between functions in different partitions). The system integrator is further responsible for guaranteeing a correct

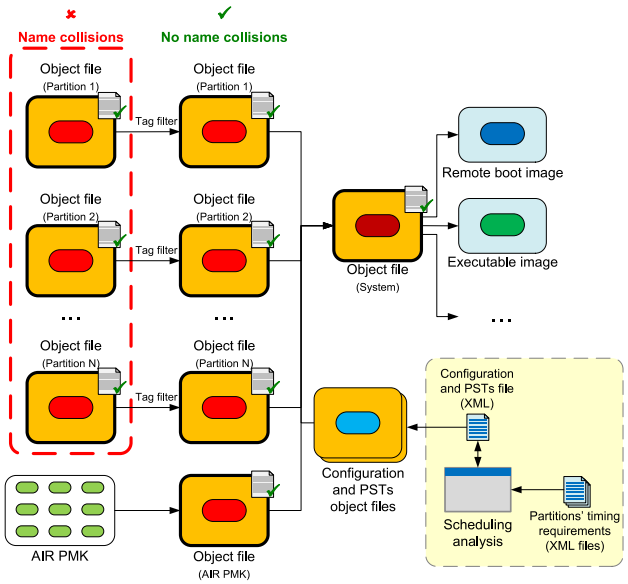

Figure 5. Integration of an AIR-based TSP system, with the introduction of TSP-specific scheduling analysis features/tools

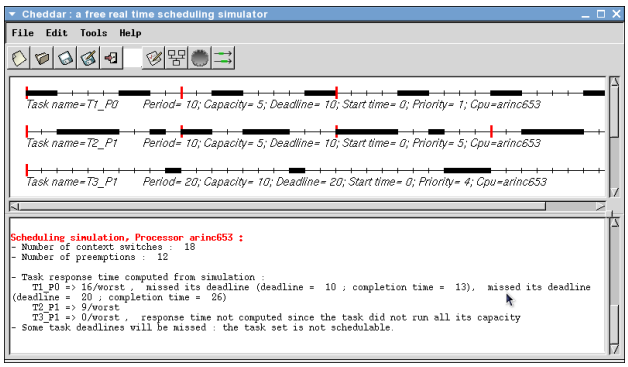

Figure 6. Cheddar scheduling tool

partition scheduling, so that partitions and the system as a whole meet their timing requisites. Thus, in the system integration phase, scheduling analysis capabilities shall be introduced in relation with the generation of a system-wide configuration [3], [23]. We will focus more on the role of schedulability analysis in system integration (in detriment of application development) for its added complexity.

A. *Cheddar: state of the art*

The necessary tool to accomplish these goals can be developed from scratch, or by extending an existing tool. We are looking into the possibility of extending Cheddar [6] 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 for either uniprocessor or multiprocessor environments.

In its latest versions, Cheddar already supports TSP schedulability analysis to some extent [6]. Figure 6 shows a Cheddar instance running a schedulability analysis on a simple TSP system. The example consists of two partitions, P0 and P1, with one and two tasks, respectively. However, Cheddar presents some limitation in its current

support for TSP schedulability analysis. A PST is defined as an array of durations. For instance, the PST for the example shown in Figure 6 is defined as:

```
partition_duration(0):=2;
partition_duration(1):=4;
```

Besides presenting less usability, the current implementation limits the PST to having only one time window per partition per MTF.

Cheddar's current support to TSP scheduling analysis allows, although not explicitly, the simulation of both multicore TSP systems involving both interpartition (Section IV-A) and intrapartition (Section IV-B) parallelism. However, besides inheriting the aforementioned limitations to PST definition, it restricts itself to approaches based on the partitioned multiprocessor real-time scheduling paradigm, and requires the assignment of partitions and/or tasks to processor cores to be performed beforehand. Cheddar's "Partition" functionality, for assigning tasks among multiple processor cores, is not supported within the scope of a TSP scheduler.

Cheddar is designed to be extensible, which reinforces the notion that it is a powerful starting point for a TSP-specific schedule analysis and generation tool. This should also take into consideration the advances in modelling, verifying and implementing ARINC 653 systems with the Architecture Analysis and Design Language (AADL) [24].

B. Requirements for TSP scheduling analysis tool

As seen, PSTs are defined in Cheddar directly among the source code of the simulated partition scheduler. For a flexible and usable TSP scheduling analysis tool, it would be ideal to support both *(i)* defining all scheduling components (partitions, partition scheduling, and processes) and respective parameters (partition cycle and duration process period and WCET, etc.) through the graphical user interface, and *(ii)* importing all the aforementioned information from files; this is the case, in Cheddar, for other supported scheduling analysis scenarios. For the TSP case, support for importing from a file should accommodate the XML schema of ARINC 653 configuration files [3].

C. Additional requirements for PST generation tool

The tool we propose should be able to aid the system integrators in constructing the PST (or PSTs) for the system. For a first approach, the tool should derive PSTs from the partitions' timing requirements (cycle and duration), receiving them as an input assumed to be correctly derived from the respective processes. The semantics of the timing requirement parameters is that, to meet the deadlines of its processes, a partition needs $duration$ units of processing time in each $cycle$-units interval.

Elaborating on this approach, the tool shall also become able to perform the previous step: obtaining each partition's timing requirements (cycle and duration) from the respective processes' characteristics.

Both these aspects should accommodate support for multicore-enabled TSP systems, observing inter- and/or intrapartition parallelism. Towards this goal, Cheddar's "Partition" feature, to assign tasks to processors, provides a solid starting point; the "Partition" feature can be extended to accommodate particularities of TSP systems, and updated to use a more recent and tighter results [13], [19], [20], [22].

VI. Conclusion

In this paper we exposed our current work on taking advantage of multicore platforms for added capacity, flexibility and safety in time- and space-partitioned (TSP) systems. The presented work encompasses architectural support to multicore in TSP systems; we use the AIR architecture for TSP systems in aerospace applications as a reference, but the functionality and mechanisms we propose apply to TSP systems in general [5]. From the real-time scheduling point of view, the crucial components of our proposal are the notions of interpartition parallelism and intrapartition parallelism [17]. A great deal of our current concerns goes to the provision of adequate tools to support the development and integration of TSP systems — both those going by the current state of the art and those which will implement the multicore facets we propose, namely inter- and intrapartition parallelism. We envision extending the Cheddar tool [6] to provide real-time scheduling analysis and partition scheduling table (PST) generation facilities for TSP systems integrators and application developers. Cheddar is a mature real-time scheduling analysis tool with limited support for TSP systems, and designed to be extended as the main building block of a full-featured schedule analysis and generation tool for TSP system development and integration.

Acknowledgement

The authors would like to thank the anonymous reviewers and the attendees of the ECRTS 2011 conference for insightful comments and discussions which allowed improving this paper, in particular Enrico Bini, Tommaso Cucinotta, Moritz Neukirchner, and Mikael Åsberg.

References

[1] J. Rushby, "Partitioning in avionics architectures: Requirements, mechanisms and assurance," SRI Int'l., California, USA, NASA Contractor Report CR-1999-209347, 1999.

[2] AEEC, "Design guidance for Integrated Modular Avionics," Aeronautical Radio, Inc., ARINC Report 651-1, Nov. 1997.

[3] ——, "Avionics application software standard interface, part 1 - required services," Aeronautical Radio, Inc., ARINC Specification 653P1-2, Mar. 2006.

[4] J. Windsor and K. Hjortnaes, "Time and space partitioning in spacecraft avionics," in *Proceedings of the 3rd IEEE International Conference on Space Mission Challenges for Information Technology*, Pasadena, CA, USA, Jul. 2009, pp. 13–20.

[5] J. Rufino, J. Craveiro, and P. Verissimo, "Architecting robustness and timeliness in a new generation of aerospace systems," in *Architecting Dependable Systems VII*, ser. Lecture Notes in Computer Science, A. Casimiro, R. de Lemos, and C. Gacek, Eds. Springer, 2010, vol. 6420, pp. 146–170.

[6] F. Singhoff, A. Plantec, P. Dissaux, and J. Legrand, "Investigating the usability of real-time scheduling theory with the Cheddar project," *Real-Time Systems*, vol. 43, no. 3, pp. 259–295, Nov. 2009.

[7] J. Craveiro and J. Rufino, "Adaptability support in time- and space-partitioned aerospace systems," in *Proceedings of the Second International Conference on Adaptive and Self-adaptive Systems and Applications (ADAPTIVE 2010)*, Lisbon, Portugal, Nov. 2010.

[8] R. Davis and A. Burns, "A survey of hard real-time scheduling for multiprocessor systems," *ACM Computing Surveys*, 2010, accepted for publication.

[9] J. Craveiro, J. Rufino, and P. Verissimo, "Time- and space-partitioned systems: History, theory and practice," AIR-II Technical Report RT-11-03, 2011, survey, in submission.

[10] I. Shin, A. Easwaran, and I. Lee, "Hierarchical scheduling framework for virtual clustering of multiprocessors," in *Proceedings of the 20th Euromicro Conference on Real-Time systems (ECRTS '08)*, Prague, Czech Republic, Jul. 2008, pp. 181 –190.

[11] E. Bini, G. Buttazzo, and M. Bertogna, "The multi supply function abstraction for multiprocessors," in *Proceedings of the 2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA '09)*, Beijing, China, Aug. 2009, pp. 294–302.

[12] E. Bini, M. Bertogna, and S. Baruah, "Virtual multiprocessor platforms: Specification and use," in *Proceedings of the 2009 30th IEEE Real-Time Systems Symposium (RTSS '09)*, Washington, D.C., USA, Nov./Dec. 2009, pp. 437–446.

[13] G. Lipari and E. Bini, "A framework for hierarchical scheduling on multiprocessors: From application requirements to run-time allocation," in *Proceedings of the 2010 31st IEEE Real-Time Systems Symposium (RTSS '10)*, San Diego, CA, USA, Nov./Dec. 2010, pp. 249–258.

[14] M. S. Mollison, J. P. Erickson, J. H. Anderson, S. K. Baruah, and J. A. Scoredos, "Mixed-criticality real-time scheduling for multicore systems," in *Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology (CIT '10)*, Bradford, UK, Jun./Jul. 2010, pp. 1864–1871.

[15] J. Craveiro, J. Rosa, and J. Rufino, "Self-adaptive scheduling of real-time applications in multicore time- and space-partitioned systems," AIR-II Technical Report RT-11-06, 2011, in submission.

[16] J. Rosa, J. Craveiro, and J. Rufino, "Safe online reconfiguration of time- and space-partitioned systems," in *Proceedings of the 9th IEEE International Conference on Industrial Informatics (INDIN 2011)*, Caparica, Lisbon, Portugal, Jul. 2011.

[17] J. Craveiro and J. Rufino, "Applying multiprocessor real-time scheduling theory to achieve parallelism in time- and space-partitioned systems," AIR-II Technical Report RT-11-01, 2011, in submission.

[18] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, pp. 46–61, January 1973.

[19] S. Lauzac, R. Melhem, and D. Mossé, "An improved rate-monotonic admission control and its applications," *IEEE Transactions on Computers*, vol. 52, no. 3, pp. 337–350, Mar. 2003.

[20] E. Bini, G. Buttazzo, and G. Buttazzo, "Rate monotonic analysis: the hyperbolic bound," *IEEE Transactions on Computers*, vol. 52, no. 7, pp. 933–942, Jul. 2003.

[21] S. K. Dhall and C. L. Liu, "On a real-time scheduling problem," *Operations Research*, vol. 26, no. 1, pp. 127–140, Jan./Feb. 1978.

[22] I. Lupu, P. Courbin, L. George, and J. Goossens, "Multi-criteria evaluation of partitioning schemes for real-time systems," in *Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2010)*, Bilbao, Spain, Sep. 2010.

[23] J. Craveiro and J. Rufino, "Schedulability analysis in partitioned systems for aerospace avionics," in *Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2010)*, Bilbao, Spain, Sep. 2010.

[24] J. Delange, L. Pautet, A. Plantec, M. Kerboeuf, F. Singhoff, and F. Kordon, "Validate, simulate, and implement ARINC653 systems using the AADL," *Ada Lett.*, vol. 29, pp. 31–44, November 2009, also published in *Proceedings of the ACM SIGAda annual international conference on Ada and related technologies (SIGAda '09)*.