Scheduling Analysis Principles and Tool for Time- and Space-Partitioned Systems

João Pedro Craveiro[1], Jeferson L. R. Souza[1], José Rufino[1], Vincent Gaudel[2], Laurent Lemarchand[2], Alain Plantec[2], Stéphane Rubini[2], and Frank Singhoff[2]

[1] Universidade de Lisboa, Faculdade de Ciências, LaSIGE (Lisbon, Portugal)
[2] Lab-STICC UMR 6285, Université de Bretagne Occidentale, UEB (Brest, France)

Abstract. This paper describes the SAPIENT (Scheduling Analysis Principles and Tool for Time- and Space-Partitioned Systems) project, whose goal is to evolve the Cheddar real-time scheduling analysis tool with the application of theoretical results for compositional hierarchical scheduling and time- and space-partitioned (TSP) systems. Besides compositional scheduling analysis, the resulting tool shall also be able to produce feasible partition scheduling tables from the timing requisites of the different applications in a TSP system.

Keywords: Cheddar, composability, compositional analysis, hierarchical scheduling, real-time systems, schedulability analysis, time and space partitioning

1 Introduction

The next generation of space vehicles integrates different mission functions on a shared computing platform, using the emerging and advanced principle of time and space partitioning (TSP). This principle implies specific scheduling considerations which no scheduling analysis tool fully integrates. TSP systems are a special case of hierarchical scheduling frameworks, a hot topic in the real-time community. In either TSP systems in specific or hierarchical scheduling frameworks in general, it is desirable to observe composability and compositionality guarantees, which allow, respectively, components to be analysed independently of the rest of the system and the system to be analysed using a view of its components which hides the internal specifics of the latter.

This paper describes the SAPIENT (Scheduling Analysis Principles and Tool for Time- and Space-Partitioned Systems) project, whose goal is to evolve the Cheddar real-time scheduling analysis tool with the application of theoretical results for compositional hierarchical scheduling and time- and space-partitioned (TSP) systems. Besides compositional scheduling analysis, the resulting tool shall also be able to produce feasible partition scheduling tables from the timing requisites of the different applications in a TSP system and generate the latter's onboard computer configuration parameters.

2 Reference architecture for multicore TSP systems

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 architecture features a modular design, with robust temporal and spatial partitioning ensured by the transversal component *Partition Management Kernel* (PMK). Temporal partitioning is achieved through a two-level hierarchical scheduling scheme. In the first level, partitions are scheduled cyclically according to a *partition scheduling table* (PST). In the second level, in each partition, processes compete according to the native process scheduler of each POS. AIR supports mode-based partition schedules, among which the system can switch for (self-)adaptation to mission changes [1,5].

Multiprocessor/multicore support To add the capacity and flexibility expected from a multicore system, we have proposed 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) [2]. The AIR PMK may take advantage of a multiprocessor or multicore platform in several ways, either in alternative or cumulatively: *(i)* interpartition parallelism; *(ii)* intrapartition parallelism; *(iii)* enhanced spatial segregation; *(iv)* fault tolerance. *Interpartition parallelism* is achieved by extending the first level of the hierarchical scheduling scheme. 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. 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 — *interpartition parallelism*. As such, a partition may parallelly execute multiple processes. Enhanced spatial segregation and fault tolerance are out of the scope of this paper.

Composability and compositionality *Composability* is the property of a component which can be integrated in different systems without additional effort or specific knowledge of the remaining system. This can be obtained with resource interfaces, which abstract to the component how computing resources are provided to it. In the AIR architecture, this results in applications (partitions) being able to be developed independently merely upon a specification of the expected resource share. *Compositionality* is the property of a system or of a component which can be analysed by knowing the results of the analysis of its subcomponents (but not their inner details) and how they are combined. This can be achieved through component interfaces, which abstract to the component or system how each of its subcomponents consumes the resources it is given [7]. In an AIR-based TSP system, this results in the system not having to change in the face of component changes as long as the latter certifiably maintain their externally observable characteristics. Abstracting processes and partitions so that their direct scheduler sees them as identical highlights the advantages of compositionality [4].

In uniprocessor, the periodic resource model (PRM) [7] can be used as a resource and component interface. A PRM $\Gamma = (\Pi, \Theta)$ abstracts the demand (as a component interface) or supply (as a resource interface) of Θ units of computing resource over every period of Π time units. Abstracting subcomponents with PRM allows using classical schedulers to schedule subcomponents as implicit-deadline periodic tasks, with an inherent reuse of decades of schedulability analysis results. For homogeneous

multiprocessors, Shin et al. have proposed the multiprocessor periodic resource (MPR) model [6]. For uniform multiprocessors, Craveiro et al. have a proposed an extension of the MPR — the heterogeneous multiprocessor periodic resource (HMPR) model [3].

3 Scheduling analysis tool for TSP systems

As described before, the observed properties of compositionality and composability allow the multiple applications that may compose such a TSP system to be developed, verified and validated independently; this eases certification efforts, since only modified modules need to be re-evaluated. 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 [2]. Developers will only have access to the component's resource interface and should be able to analyse their application's feasibility upon it using compositional analysis. Further in the development process, the applications are integrated into the system; the system integrator is further responsible for guaranteeing a correct 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 [1, 2].

Cheddar state of the art Cheddar [8] is an opensource toolset composed of a graphical editor and a library of processing modules. It aims at providing performance analysis of concurrent real-time applications. Its library implements several classical real-time scheduling algorithms and schedulability tests, based on a simple architecture description language (ADL). The Cheddar ADL defines basic entities, which are reusable units, divided in two types: hardware components (e.g. processors, memories) and software components, which represent the design of the software (e.g. tasks, address spaces). Nevertheless, schedulers are modelled as an attribute of a processor or an address space. This limits the scope of representable architectures. Time and space partitioning is restricted to the partitioned multiprocessor real-time scheduling paradigm, with a static assignment of partitions and tasks which has to be performed by hand. Thus, Cheddar only allows the modelling and analysis of two-level hierarchical schedulers [2].

Partition scheduling table generation As seen, PSTs are defined in Cheddar directly among the source code of the simulated partition scheduler [2]. 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 (component interfaces, tasks) 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. The tool we propose should be able to aid the system integrators in constructing the PST (or PSTs) for the system. As a first step, we are exploiting the PST generation problem for TSP systems observing interpartition parallelism over homogeneous multiprocessor platforms. We furthermore assume we have each partition's timing requirements correctly obtained from those of its processes and expressed as a PRM (Π, Θ) and generate the PST from those.

Schedulability analysis and component abstraction As stated in [2], current TSP schedulability analysis in Cheddar presents some limitations (partition statically assigned to one processor, heterogeneous description model of hierarchical scheduler, constant PSTs). To extend the usage scope of Cheddar for the verification of multicore-aware TSP systems, an evolution of its ADL is necessary. Its new ADL will be structured by two basic entity sets: component entities and binding entities. The binding entities are a new feature we append in order to represent the relationships between a set of components and another set of components. The relation should be interpreted as the permission, for a set of components to allocate a set of shared resources More precisely, two types of bindings will be defined. *(i) Spatial bindings* specify what are the storage and computing resources that tasks or address spaces may use (i.e. the software deployment on hardware platforms). *(ii) Temporal bindings* determine the dynamic allocation of a set of resources onto a set of tasks at any instant. Applied to a TSP multiprocessor system, a task and an address space model a partition. The spatial binding allows the partition to be deployed on hardware components. A first level scheduler establishes the temporal binding of the partitions onto the processing units, following the scope of resources specified by the spatial binding. Next, in-partition resource sharing leans on the same principles, except that hardware resources are not processors or memories, but first level tasks or address spaces. The Cheddar tool shall become able to obtain each partition's timing requirements from the respective address spaces' characteristics employing the compositional approach (see Sect. 2).

This work was partially supported by FCT/Égide (PESSOA programme), through the transnational cooperation project SAPIENT. This work was partially supported by the EC, through project IST-FP7-STREP-288195 (KARYON). This work was partially supported by FCT, through the Multiannual and CMU|Portugal programs, and the Doctoral Grants SFRH/BD/60193/2009 and SFRH/BD/45270/2008.

References

1. AEEC: Avionics application software standard interface, part 1 - required services. ARINC Spec. 653P1-2 (Mar 2006)
2. Craveiro, J., Rufino, J., Singhoff, F.: Architecture, mechanisms and scheduling analysis tool for multicore time- and space-partitioned systems. ACM SIGBED Rev. 8(3), 23–27 (2011)
3. Craveiro, J.P., Rufino, J.: Heterogeneous multiprocessor compositional real-time scheduling. In: RTSOPS 2012. Pisa, Italy (Jul 2012)
4. Craveiro, J.P., Silveira, R.O., Rufino, J.: hsSim: an extensible interoperable object-oriented n-level hierarchical scheduling simulator. In: WATERS 2012. Pisa, Italy (Jul 2012)
5. Rufino, J., Craveiro, J., Verissimo, P.: Architecting robustness and timeliness in a new generation of aerospace systems. In: Casimiro, A., de Lemos, R., Gacek, C. (eds.) Architecting Dependable Systems VII, LNCS, vol. 6420, pp. 146–170. Springer (Nov 2010)
6. Shin, I., Easwaran, A., Lee, I.: Hierarchical scheduling framework for virtual clustering of multiprocessors. In: ECRTS '08. Prague, Czech Republic (Jul 2008)
7. Shin, I., Lee, I.: Periodic resource model for compositional real-time guarantees. In: RTSS '03. Cancun, Mexico (Dec 2003)
8. Singhoff, F., Plantec, A., Dissaux, P., Legrand, J.: Investigating the usability of real-time scheduling theory with the Cheddar project. Real-Time Systems 43(3), 259–295 (Nov 2009)