Open Source Implementation of Hierarchical Scheduling for Integrated Modular Avionics *

Juan Zamorano, Juan A. de la Puente
Universidad Politécnica de Madrid (UPM)
E-28040 Madrid, Spain
jzamora@fi.upm.es, jpuente@dit.upm.es

Alfons Crespo
Universidad Politécnica de Valencia (UPV)
E-46022 Valencia, Spain
alfons@disca.upv.es

Abstract

This paper describes the porting of a Ravenscar compliant kernel (ORK+) to the hypervisor XtratuM to build up an open source ARINC 653 platform for avionics systems. The Integrated Modular Avionics (IMA) architecture requires a specialized operating system layer that provides temporal and spatial isolation between partitions. The ARINC 653 standard defines an architecture and an applications program interface (API) for such an operating system or *application executive* (APEX), in ARINC terms. There are diverse ARINC 653 implementations available from multiple vendors, and the standard has been successfully used in a number of commercial and military avionics systems. However, there was not an open source ARINC 653 platform available. The combination of both tools (ORK+ and XtratuM) provides an IMA platform that allows different criticality applications to share the same computer board.

1 Introduction

Current trends in avionic systems envisage systems with increased functionality and complexity. Such systems are often composed of several applications that may have different levels of criticality. In such a scenario, the most critical applications must be isolated from the less critical ones, so that the integrity of the former is not compromised by failures occurring in the latter. Isolation has often been achieved by using a *federated* approach, i.e. by allocating different applications to different computers. However, the growth in the number of applications and the increasing processing power of embedded computers foster an *integrated* approach, in which several applications may be executed on a single computer platform. In this case, alternate mechanisms must be put in place in order to isolate applications from each other. The common approach is to provide a number of *logical partitions* on each computer platform, in such a way that each partition is allocated a share of processor time, memory space, and other resources. Partitions are thus isolated from each other both in the temporal and spatial domains. Temporal isolation implies that a partition does not use more processor time than allocated, and spatial isolation means that software running in a partition does not read or write into memory space allocated to another partition.

This approach has been successfully implemented in the aeronautics domain by so-called Integrated Modular Avionics (IMA) [10]. The IMA architecture requires a specialized operating system layer that provides temporal and spatial isolation be-

*This work has been partly funded by the Spanish Ministry of Science, project TIN2008-06766-C03-01 (RT-MODEL)

tween partitions. The ARINC 653 standard defines an architecture and an applications program interface (API) for such an operating system or *application executive* (APEX), in ARINC terms. Temporal isolation is provided by using a two-level scheduling scheme. A *partition scheduler* allocates processor time to partitions according to a static cyclic schedule, where each partition runs in turn for the duration of a fixed slice of time (cf. figure 1). The ARINC global scheduler is a variant of a static cyclic executive, while the local schedulers are priority-based. Spatial isolation between partitions is provided by implementing a separate address space for each partition, in a similar way as process address spaces are protected from each other in conventional operating systems. There are diverse ARINC 653 implementations available from multiple vendors, and the standard has been successfully used in a number of commercial and military avionics systems. However, there was not an open source ARINC 653 platform available.

This paper describes the porting of an improved version of the Open Ravenscar Kernel (ORK+) [11] to the hypervisor XtratuM [9] to build up an open source ARINC 653 platform [3] for avionics systems. The combination of both tools provides an IMA platform that allows different criticality applications to share the same computer board. Section 2 describes the main features of the hypervisor XtratuM. Section 3 details the architecture of ORK+ and changes needed to port it to XtratuM. Finally, some conclusions about the resulting ARINC 653 complaint operating system are drawn and plans for the near future are explained in section 4.

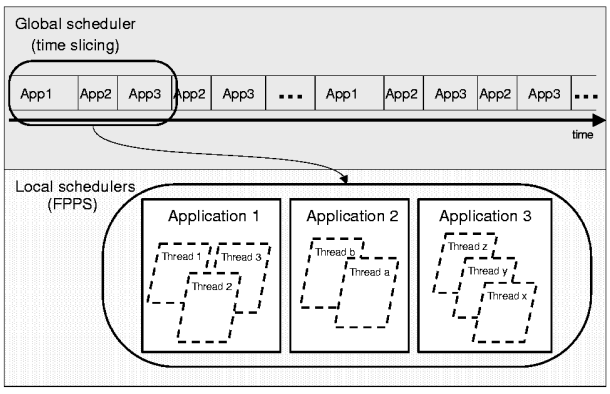

FIGURE 1: *ARINC 653 hierarchical architecture.*

2 XtratuM Overview

XtratuM [7, 8, 6] is a type 1 hypervisor that uses para-virtualisation. It was designed to meet safety critical real-time requirements. The most relevant features are:

- Bare hypervisor.
- Employs para-virtualisation techniques.
- An hypervisor designed for embedded systems: some devices can be directly managed by a designated partition.
- Strong temporal isolation: fixed cyclic scheduler.
- Strong spatial isolation: all partitions are executed in processor user mode, and do not share memory.
- Fine grain hardware resource allocation via a configuration file.
- Robust communication mechanisms (ARINC653 sampling and queuing ports).

XtratuM provides a virtual machine closer to the native hardware. The virtual machine provides the access to the system resources: cpu registers, clock, timer, memory, interrupts, etc., through a set of system calls (hypercalls). Each virtual machine is referred as "guest", "domain" or "partition" and can contain a bare code or an operating system with the applications. To execute an operating system as a guest partition, it has to be para-virtualised which implies the replacement of some parts of the operating system HAL (Hardware Abstraction Layer) with the corresponding hypercalls.

In this approach, a partition is a *virtual computer* rather than a group of strongly isolated processes. When multi-threading (or tasking) support is needed in a partition, then an operating system or a run-time support library has to provide it.

Figure 2 shows the XtratuM architecture and the relation with the partitions.

The services provided by XtratuM are summarised in the table 1. Additionally to these general services, XtratuM provides specific services that strongly depend on the native processor that are shown in table 2.

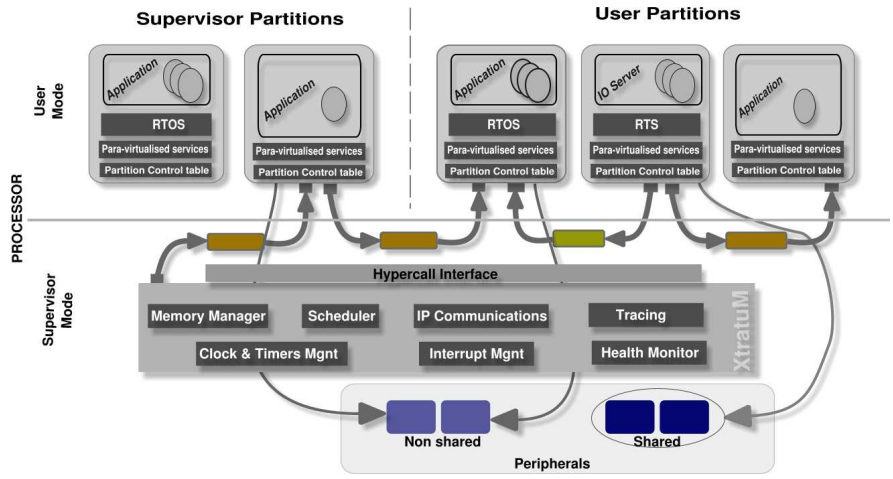

FIGURE 2: *XtratuM and its development environments.*

Group of services	Hypercalls	Partition type
Clock management	get clock; define timers	Normal
IRQ Management	enable / disable IRQs, mask / unmaks IRQs	Normal
IP Communication	create ports; read / receive / write / send messages	Normal
IO management	read/write IO	Normal
Partition management	mode change, halt / reset / resume / suspend / shutdown partitions (system)	System
Health monitoring management	read / seek / status HM events	System
Audit facilities	read / status	System

TABLE 1: *XtratuM hypercalls*

SparcV8 hypercalls
XM_sparcv8_atomic_add
XM_sparcv8_atomic_and
XM_sparcv8_atomic_or
XM_sparcv8_ush_regwin
XM_sparcv8_get_ags
XM_sparcv8_inport
XM_sparcv8_iret
XM_sparcv8_outport
XM_sparcv8_set_ags

TABLE 2: *XtratuM SparcV8 hypercalls*

Currently, version 2.2 is being used by CNES as a TSP-based solution for building highly generic and reusable on-board payload software for space applications [1]. TSP (time and space partitioning) based architecture has been identified as the best solution to ease and secure reuse. It enables a major decoupling of the generic features that are being developed, validated, and maintained in mission-specific data processing [2].

3 ORK+ Overview

ORK [4] is an open-source real-time kernel which provides full conformance with the Ravenscar tasking profile on embedded computers. The kernel has a reduced size and complexity, and has been carefully designed to allow the building of reliable software for embedded applications. This kernel is integrated in a cross-compilation system based on GNAT, supporting the subset of Ada tasking which is allowed by the Ravenscar profile in an efficient and compact way.

ORK+ [11] includes support for the new Ada 2005 timing features, such as execution time clocks and timers. The ORK+ kernel provides all the required functionality to support real-time programming on top of the LEON2 hardware architecture. The kernel functions are grouped as follows:

1. Task management, including task creation, synchronization, and scheduling.

2. Time services, including absolute delays and real-time clock.

3. Interrupt handling.

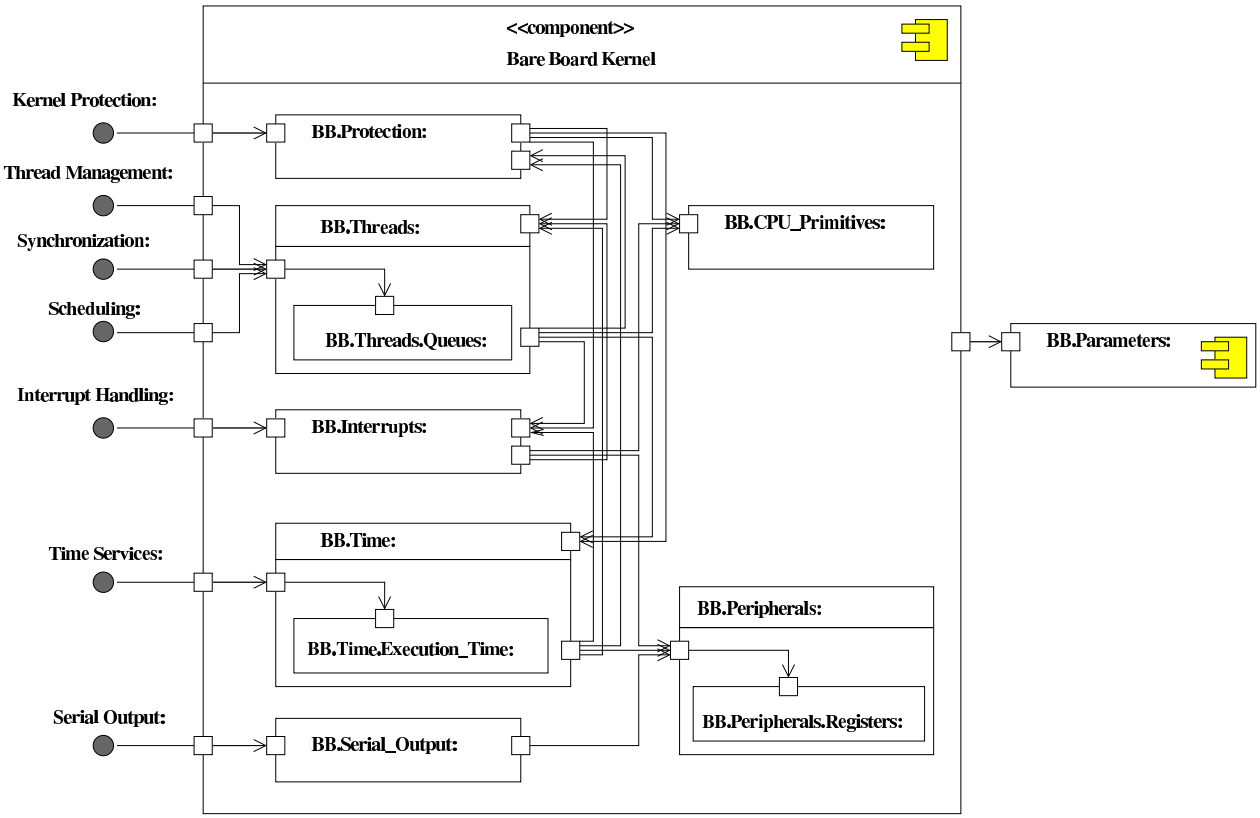

FIGURE 3: *ORK+ architecture.*

That kernel functionality is implemented with the following components (figure 3):

- `System.BB`:[1] Root package (empty interface).
- `System.BB.Threads`: Thread management, including synchronization and scheduling control functions.
- `System.BB.Threads.Queues`: Different kernel queues such as alarm queue and ready queue.
- `System.BB.Time`: Clock and delay services including support for timing events.
- `System.BB.Time.Execution_Time_Support`: Execution time clocks and timers as well as group budgets support.
- `System.BB.Interrupts`: Interrupt handling.
- `System.BB.Parameters`: Configuration parameters.
- `System.BB.CPU_Primitives`: Processor-dependent definitions and operations.
- `System.BB.Peripherals`: Support for peripherals in the target board.
- `System.BB.Peripherals.Registers`: Definitions related to input-output registers of the peripheral devices.
- `System.BB.Serial_Output`: Support for serial output to a console.

That software architecture make easy to port ORK+ to different hardware or software platforms. Most of kernel components are fully independent from the underlying platform and the lower level components have specifications that are also platform independent. In this way, it is only needed to re-implement the lowest level functions on top of the provided XtratuM functions to obtain the full ORK+ functionality.

The modified components were `System.BB.Time`, `System.BB.Interrupts`, `System.BB.CPU_Primitives`, `System.BB.Time.Execution_Time_Support` and `System.BB.Peripherals`.

[1] "BB" stands for "bare-board kernel".

4 Conclusions and future work

The combination of both tools builds up an ARINC 653 complaint operating system providing an IMA platform that allows different criticality applications to share the same computer board. Those applications can be developed in an independent way and the ARINC 653 platform provides at run-time their corresponding time slots and memory in a safe and secure way.

Applications can be developed by either using ORK+ or XtratuM primitives to support threads and have timing facilities among others. However, ORK+ provides an additional level of temporal isolation by means of execution time clocks and timers. In this way, it is possible to bound the execution time of threads in such a way that they are not allowed to execute longer than previously calculated.

XtratuM and ORK+ are currently targeted to LEON2 [5] based computers[2]. LEON2 has not Memory Management Unit but a reduced set of fence registers, therefore the spatial isolation is not full because LEON2 fence registers do not have protection against read operations. Moreover, that limited memory protection mechanism also imposed a rigid memory sharing scheme between different partitions. These limitations can be overcome with the next-generation of LEON processors, LEON3, supporting full-featured MMUs. We plan to port the whole system to LEON3 platforms as well to other standard hardware such as the ix86 PC compatible architecture. This will enable it to be used in other application domains, and also in real-time systems education.

References

[1] P. Arberet, J.-J. Metge, O. Gras, and A. Crespo. TSP-based generic payload on-board software. In *DASIA 2009. DAta Systems In Aerospace.*, May. Istanbul 2009.

[2] P. Arberet and J. Miro. IMA for space : status and considerations. In *ERTS 2008. Embedded Real-Time Software.*, Jannuary. Toulouse. France 2008.

[3] ARINC. *Avionics Application Software Standard Interface — ARINC Specification 653-1*, October 2003.

[4] Juan A. de la Puente, José F. Ruiz, and Juan Zamorano. An open Ravenscar real-time kernel for GNAT. In Hubert B. Keller and Erhard Plödereder, editors, *Reliable Software Technologies — Ada-Europe 2000*, number 1845 in LNCS, pages 5–15. Springer-Verlag, 2000.

[5] Gaisler Research. *LEON2 Processor User's Manual*, 2005.

[6] M. Masmano, I. Ripoll, A. Crespo, and J.J. Metge. Xtratum: a hypervisor for safety critical embedded systems. In *Eleventh Real-Time Linux Workshop*, 2009.

[7] M. Masmano, I. Ripoll, A. Crespo, J.J. Metge, and P. Arberet. Xtratum: An open source hypervisor for TSP embedded systems in aerospace. In *DASIA 2009. DAta Systems In Aerospace.*, May. Istanbul 2009.

[8] M. Masmano, I. Ripoll, S. Peiró, and A. Crespo. Xtratum for leon3: an open source hypervisor for high integrity systems. In *European Conference on Embedded Real Time Software and Systems. ERTS2 2010.*, 2010.

[9] Miguel Masmano, Ismael Ripoll, Alfons Crespo, and Jean-Jacques Metge. Xtratum: a hypervisor for safety critical embedded systems. In *11th Real-Time Linux Workshop*, Dresden. Germany., 2009.

[10] John Rushby. Partitioning for safety and security: Requirements, mechanisms, and assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999. Also to be issued by the FAA.

[11] Santiago Urueña, José Antonio Pulido, José Redondo, and Juan Zamorano. Implementing the new Ada 2005 real-time features on a bare board kernel. *Ada Letters*, XXVII(2):61–66, August 2007. Proceedings of the 13th International Real-Time Ada Workshop (IRTAW 2007).

[2]LEON2 is a radiation-hardened implementation of the SPARC V8 architecture, which has been adopted by the European Space Agency (ESA) as the new standard processor for spacecraft on-board computer systems.