Validation of securely partitioned systems over multicore architectures based on Xtratum

Javier Coronel[1], M. Tsagkaropoulos[2], Dimitrios Mylonas[2], Patricia Balbastre[3], Vangelis Kollias[2] and Alfons Crespo[3]

[1] *FentISS S.L., Edificio 9B Despacho 3 Camino de Vera s/n, 46022 Valencia, Spain, Tel: +34 963 294 704, Email: jcoronel@fentiss.com*

[2] *Teletel S.A. 124 Kifissias Avenue, 11526, Athens, Greece, Tel: +30 210 6983393, Fax: +30 210 6983391, Email: m.tsagkaropoulos@teletel.eu*

[3] *Instituto de Automática e Informática Industrial (ai2), Universitat Politècnica de València. Camino de Vera s/n, 46022 Valencia, Spain, Tel: +34 96 387 5707, Email: patricia@ai2.upv.es*

1. ABSTRACT

Growing complexity of embedded systems with increased functionality while at the same time pushing for less power consumption, less processor needs, smaller memory footprints and less weight, makes the integration of security and safety an issue in many domains (e.g. automotive, railway, automation control and aerospace). In order to follow these technology tendencies, there is a growing interest in the supporting of mixed criticality for embedded systems based on multicore by adopting virtualisation technology.

The present survey is a study of the porting of XtratuM hypervisor over multicore architectures, complemented with the supporting validation methodology toward the verification and validation of the multicore partitioned system.

2. INTRODUCTION

Historically many safety-related and security-critical systems have been developed and qualified using single-core processors. These single-core platforms could easily meet higher performance requirements by increasing the processor clock speed. However, the industry is now approaching the limit of this relatively simple upgrade path and this is creating an increasing trend towards adoption of multicore processor architectures in critical systems to meet the demand for higher performance. There is a number of challenges involved in the migration to multicore processor architectures in safety-critical and security-sensitive embedded systems domain which are still unresolved and which contribute to increase the complexity related to the development.

One way to avoid the increased validation and certification effort in safety-critical systems is to incorporate mechanisms that establish multiple partitions on the same hardware platform with strict temporal and spatial separation between the individual partitions. In this approach, applications with different levels of criticality can be placed in different partitions and can be verified and validated in isolation.

Virtualisation technologies can provide huge improvements in flexibility, cost and time to market of safety-critical and secure embedded applications and can provide partitioning. Furthermore, multicores offer better performance than single-core processors, while maintaining a relatively simple processor design.

ESA project launched a project on multicore evaluation for its use in space applications [3]. This project tried to evaluate the suitability of LEON4 multicore processor in partitioned developments. XtratuM was selected to be ported on top of this architecture.

XtratuM [1] is a hypervisor for embedded real-time systems that initially was developed for x86 processor and ported to LEON2 and LEON3. XtratuM uses para-virtualisation technique to build a virtualisation layer.

In [2] a preliminary study and implementation of XtratuM over a LEON4 was proposed. The aim of this work is to present a detailed design, as well as a complete validation of the multicore virtualisation layer.

As stated in [2], for the design of the multicore virtualisation layer, we propose to use a hybrid solution based on a SMP hypervisor layer that provides virtualisation services to guest applications. Each application can run its own OS on its virtual processor. The hypervisor kernel is able to enforce a strict time and space partitioning of the hardware resources.

This approach has the advantages of allowing a smooth transition from monocore applications to multicore platforms. An application developed to run on top of a mono- core hypervisor would run transparently on a multicore hypervisor, only the hypervisor kernel needs to be ported and qualified to the multicore processor.

3. XTRATUM DESIGN FOR MULTICORE

In multicore approach, the hypervisor can provide several virtual CPUs to the partitions. A partition can be mono or multi core. Different partitions (from the number of cores point of view) can coexist in the system. It allows to profit from a multicore platform even if the partitions are not multicore. Figure 1 shows the multicore approach followed for the implementation of the hypervisor.

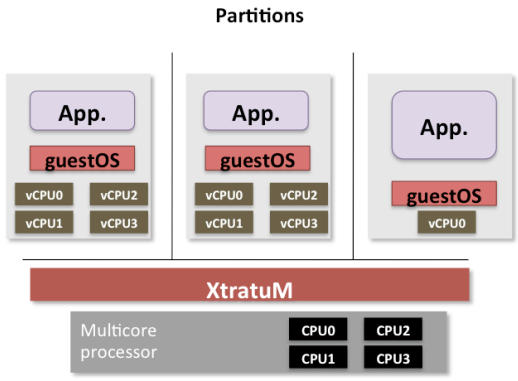

Figure 1: Multicore approach

XtratuM provides virtualisation services to partitions. It is executed in supervisor processor mode and virtualises the CPU, memory, interrupts, and some specific peripherals. XtratuM functionality includes:

- Memory management.
- Scheduling.
- Interrupt management.
- Clock and timer management.
- Inter-partition communication.
- Health monitor.

Figure 2 represents the XtratuM class.

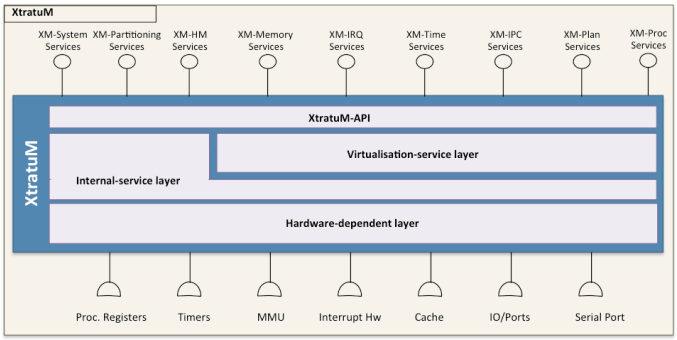

Figure 2: Virtualisation layer architecture

In this diagram the following element are considered:

- **Hardware-dependent layer**: represents the class of objects that deal with the hardware elements. It provides the Hardware-dependent layer implementation as a set of drivers required to manage the strictly necessary hardware: processor, interrupts, hardware clocks, hardware timers, paging, etc. This layer is isolated from the rest through the Hardware Abstraction Layer (HAL). Thus, the HAL hides the complexity of the underlying hardware by offering a high-level abstraction of it.
- **Internal-service layer**: It provides the internal supporting services. These services are not available to the partitions. This layer includes a minimal C library (KLibC) which provides the strictly required set of standard C functions (e.g. strcpy, memcpy, sprintf) and a bundle of data structures (e.g. a list, a heap and a queue). The boot code is also part of the internal services.
- **Virtualisation-service layer**: It provides the services required to support the virtualisation and the para-virtualisation services, which are provided via the hypercall mechanism to partitions. Some of these services are also used from other XtratuM modules. For example, the physical memory manager is in charge of allocating free physical memory pages to both, XtratuM and partitions. The internal modules are:
  - Guest manager: partition creation, deletion, suspension, etc. This module is also in charge of the guest system table and the guest control page.
  - Inter-Partition Communication. That is in charge of the Inter-partition communications.
  - Scheduler: is responsible for scheduling partitions.
  - Interrupt manager: handles hardware interrupts and traps. It is also in charge of triggering virtual interrupt and traps to partitions.
  - Hypercall dispatcher: handles the synchronous calls made from a partition to XtratuM, analogous to the use of system calls in conventional operating systems.
  - System clock: provides a nanosecond resolution clock to the system. In addition, it implements two clocks per partition (reachable both of them via a hypercall): a virtual clock which only advances while the partition is executing and a real one counting since machine boot.
  - Vtimer: implements nanosecond resolution virtual timers. Like the system clock, this module also offers two virtual timers per partition: a virtual timer only active while the partition is executing and a real one, always active.
  - Virtual memory manager: is in charge of managing the virtual maps, being able to create/release them and map/unmap physical pages.
  - Physical memory manager: is responsible for managing the system's physical memory pages: firstly, keeping track of the state of any physical page (e.g. type, owner, ...) and, secondly, behaving as a physical page allocator for the partitions and XtratuM itself.
  - The Output console: holds all the text messages printed by XtratuM or the partition debugging messages. The output server can redirected its content to an output hardware device.
- **XtratuM-API**: It corresponds to the partition support services. It provides a API to use the services provided by XtratuM. The library `libxm` contains the wrappers for the XtratuM services.

There are two basic communication mechanisms between the partition and the hypervisor: hypercalls and shared memory.

4. VALIDATION METHODOLOGY

A key requirement of modern multicore embedded systems to be used in added-value has traditionally been if security (and especially if multiple levels of security) and safety considerations are required. With the introduction of a secure separation kernel and hypervisor, the traditional OSs or RTOS and applications can be run in their own unclassified partition, thus allowing for the functionality of a known user interface and applications, without compromising the security of the rest of the system.

According to the TSP Working group [4], in the Avionics Time and Space partitioning User Needs, a set of potential benefits for space avionics is identified for secure partitioning microkernels:

- Preventing failures in a certain function of the software to propagate to other functions.
- Avoid the typical situation where a small change in one function can result in having to do extensive non-regression testing.
- Less effort and risk for integration of functional chains.

The above benefits can be also extended to other industry sectors, e.g. automotive, automation etc., where the safety of the system and the conformance to a security profile is needed. These can be achieved by using the concepts of Time and Space partitioning and secure partitioning.

Towards this direction, a generic test suite is defined, that can be used by microkernels developers, as well as developers of applications that are to be deployed over TSP architectures, like the Xtratum architecture, to assist them in testing and validating the virtualisation layer and the whole system using the virtualisation layer according to the functional and non-functional requirements.

The basic categories for the test and validation of such multicore virtualized environments are divided as follows: safety validation, security validation and performance evaluation

The initial description of the main validation areas which have been defined are presented below.

4.1 Safety Validation

The safety validation tests aim at verifying that an O/S kernel achieves a desired level of safety within a partitioned architecture in the frame of real-time embedded systems, especially for the sector of Integrated Modular Avionics (IMA) [5]. The basis for these tests is derived from the ARINC-653 specification [6], which verify the compliance of an O/S kernel to a service API defined by the specification. ARINC-653 standardizes an Application Executive (APEX) Interface and the functionality that should be offered by an O/S through it as a set of services provided to safety critical applications.

There are six basic categories of services defined by the standard aiming at ensuring that the conformant microkernel provides the desired level of safety. These are the following:

- Partition Management
- Process Management
- Time Management
- Inter-Partition Management
- Intra-Partition Management
- Health Monitoring

Safety is achieved through the satisfaction of two main characteristics: dependability related to the first five services (i.e. spatial and temporal isolation) and fault tolerance (i.e. health monitoring).

The conformance test suite based on ARINC-653 regards the intended behaviours of all API services of the multicore virtualisation layer as the elements under test aiming at the verification of (a) the expected behaviour and (b) the expected results.

Each assertion is the basis for one or more test cases. Test cases will be implemented by test procedures which fall in one of the two following categories, according to the conditions under which the test is to be performed.

- Service Functional Test Procedures: examines one intended behaviour each time and its expected results under normal conditions aiming at verifying the correct implementation of a service.
- Service Robustness Test Procedures: examines the behaviour of a service used in an erroneous way by use of error injection methods or stimulation of specific conditions to verify the ability of the O/S to handle certain kinds of errors and react as predetermined.

4.2 Security Validation

In order to address the security functional and assurance requirements for a class of separation kernels, the Separation Kernel Protection Profile (SKPP) [7] has been defined. The purpose of the SKPP is to provide a Common Criteria protection profile for high assurance separation kernels.

Unlike those traditional security kernels which perform all trusted functions for a secure operating system, a separation kernel's primary security function is to partition (separate) the subjects and resources of a system into security policy-equivalence classes, and to enforce the rules for authorized information flows between and within partitions. The SKPP prescribes many measures aimed at increasing security so that the kernel's functionality be simplified and limited to separating resources to prevent subjects in one partition from interacting with subjects in other partitions.

A set of test cases based on the SKPP profile concerning security has been defined for the validation of XtratuM under the following categories:

- Security Audit

- User Data Protection
- Security Management
- Protection of the TSF
- Hypervisor Stack Leaks
- Hypervisor Slot Overrun
- Corruption of application data space and control
- Corruption of the service API parameters.

Each test case has a certain objective and covers a subset of security requirements with regards to one kind of attack among (Privilege (partition type), Confidentiality, Integrity, DoS, Safety).

4.3 Performance Analysis

Typically, an embedded system is a special-purpose information processing system that is closely integrated into its environment. It is usually dedicated to a certain application domain and knowledge about the system behaviour at design time can be used to minimize resources while maximizing predictability. In addition, the functional and non-functional properties of the whole system not only depend on the computations inside the various nodes but also on the interaction of the various data streams on the common communication media.

In this concept, a set of performance evaluation test cases has been defined which will provide performance related metrics for the virtualisation framework and the applications executed on top of it. The performance metrics include timing analysis, i.e. execution time calculations and validation of time constraints, and conformance with non-functional requirements.

Moreover, the budget overhead brought by the XtratuM virtualisation framework will be estimated and additional overheads that affect the performance of the target system including:

- Partition context switch time overhead.
- Inter-partition communication overhead.
- Overhead due to memory protection mechanism (Address translation / verification…).
- Evaluation of the memory overhead due to the partitioning (duplication of code into the partition, etc.).
- Performance degradation due to concurrent access to system resources (e.g. cache, devices etc.), Memory controller and Data bus.

5. CONCLUSIONS

This survey presents an overview of virtualisation technology for partition embedded systems and the need to support multicore architectures. In this concept, the XtratuM hypervisor, which was initially developed for real-time embedded systems with monocore x86 and LEON2/3 processors, is ported over multicore architectures. The proposed design of the XtratuM multicore virtualisation layer provides virtualisation services to guest applications which can run on their virtual processor enforcing a strict time and space partitioning of the hardware resources.

In order to avoid the increased validation and certification effort the supporting verification and validation methodology of multicore safety-critical partitioned system is analysed. The validation methodology, as presented in this survey, provides the different aspects of validation in terms of security, safety and performance, which will enable the end users to evaluate the fulfilment of the mixed-criticality software validation and verification requirements in their implementations of Time and Space Partitioning architectures with multicore CPUs at the maximum possible extend.

6. ACKNOWLEDGEMENTS

The work has been partially funded by the project COBAMI: Mission-based Hierarchical Control. Education and Science Department, Spanish Government. CICYT: MICINN: DPI2011-28507-C02-01/02 and the project MULTIPARTES of the European Commission FP7.

7. REFERENCES

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

[2] M. Patte, A. Crespo, M. Zulianello, V. Lenfftz, M. Masmano, J. Coronel. “System impact of distributed multicore systems”, In Data Systems In Aerospace (DASIA) 2012.

[3] ESA project. Integrated modular avionics for space (ima-sp), 2010-2012.

[4] TSP Working Group, “Avionics time and space partitioning user needs,” Technical Note TEC-SW/09-247/JW, Aug. 2009, ESA–ESTEC.

[5] RTCA DO-297/EUROCAE ED-124, “Integrated Modular Avionics (IMA) development guidance and certification considerations”, 2005.

[6] Airlines electronic engineering committee (AEEC), avionics application software standard interface - ARINC specification 653 - part 1 (supplement 2 - required services), 2006, ARINC Inc.

[7] U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness, Version 1.03, Information Assurance Directorate, National Security Agency, 29 June 2007.