Benchmarking Analysis and Characterization of Hypervisors for Space Multi-core Systems

Vittoriano Muttillo *, Luca Tiberi † and Luigi Pomante ‡
Università degli Studi dell'Aquila, L'Aquila, Italy 67100

Paolo Serri §
Thales Alenia Space, L'Aquila, Italy

Multi-core systems are becoming widely diffused in the space domain, mainly because of the opportunities to improve performance and, at the same time, to optimize orthogonal metrics (e.g., cost, power consumption, etc.). However, the need to consider Safety Integrity Levels (SILs), dictated by the relevant standards (i.e., ECSS-Q-ST-80C, ECSSQ-ST-30, ECSSQ-ST-40C, NPD 8700.1E, NPR 8715.3D, JMR-001, OST 134-1021-99), into space applications adds further challenges. In fact, in a multi-core system, multiple processing cores share various resources, therefore, the implementation of modern and complex Integrated Modular Avionics (IMA) systems is problematic. Actually, IMA systems are inherently mixed-critical ones where spatial and temporal isolation in the shared resources access are essential. As a way to mitigate such a problem, the exploitation of virtualization technologies allows to guarantee isolation and certification requirements but, at the same time, introduces scheduling overhead and new issues. The selection of the best virtualization technology is just one part of the solution. Then, further benchmarking and validation activities should help developers to certify virtualized environments running on multi-core architectures. In such a context, this work proposes a methodology for characterisation of hypervisor technologies in the space domain. Experimental results obtained by applying the proposed methodology in order to characterize some reference hypervisors features closes the paper.

I. Introduction

The advent of multi-core processors have opened the possibility to introduce new technologies in the embedded system domain. Regarding space domain, multi-core processors will be the european reference technology in the realization of the next-generation mini and micro satellites, exploiting the *European Space Agency* (ESA) *Next Generation Microprocessor* (NGMP) as the reference platform [1]. Multi-core architectures offer many advantages, since they allow

*Post-Doc Researcher, Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Via Vetoio, vittoriano.muttilllo@univaq.it.
†Research Fellow, Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Via Vetoio, luca.tiberi@graduate.univaq.it.
‡Assistant Professor, Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Via Vetoio, luigi.pomante@univaq.it.
§R&D Embedded Software Engineer, Thales Alenia Space Italia, Via Campo di Pile, paolo.serri@thalesaleniaspace.com.

high performance with lower clock frequency, reduced energy consumption and reduced size, with respect to a classic multi-processor approach (i.e., the use of multiple mono-core processors). For this, multi-core systems are currently widely used, both for generic processing systems and application-specific ones. However, their use is still limited in contexts where reliability and predictability are strict requirements and a certification is mandatory. As discussed in [2], the use of multi-core processing platforms poses several issues with respect to certification, both from the hardware and software point of view (e.g., shared cache, peripherals, and memory controller certification issues). For this, the avionic and space domains are particularly interesting scenarios.

Historically, the traditional approach in avionic/space domains considers *Federated Architectures* (FA), where dedicated computing resources implement single isolated functionality. In this way, the interference among them is very limited since they are quite completely enclosed into isolated components. However, with such an approach, the increase of systems complexity and the related number of functionality pose additional challenges in terms of cost, power consumption and timing. This is why industries have moved from FA to the so-called *Integrated Modular Avionics* (IMA) approach, where different functionality are implemented on the same shared computing platform. Moreover, different applications running on a IMA platform can have different safety levels. So, this kind of systems are inherently also *Mixed-Criticality Systems* (MCSs) [3]. Since determinism and isolation must be guaranteed, it is necessary to ensure a rigorous spatial and temporal isolation among the various SW components, according to one or more of the relevant domain-dependent standards (e.g., ARINC 653 [4–7]). There are several well-known techniques for the implementation of partitioned systems (e.g., dedicated separation kernels [8]) and several IMA MCS based on single-core implementations have been already qualified and used in certified commercial aircrafts domain. However, in recent years, there have been also various proposals for the implementation of such systems on multi-core processors by mainly exploiting kernels supporting *Simmetric Multi-Processing* (SMP) like *RTEMS* [9], deterministic hardware architectures [10], or virtualization technologies based on *Hypervisors* (HPV) like *Xtratum* [11], *PikeOS* [12], Wind River Hypervisor [13], INTEGRITY Multivisor [14], Virtuosity OA [15], etc. In particular, virtualization is a technique for implementing robust IMA platforms based on multi-core processors because it allows to run multiple applications with different safety levels into isolated SW partitions on top of an hypervisor able to manage real-time scheduling and the access to shared HW resources.

In this context, this work focuses on the benchmarking analysis of multi-core processor architectures that exploit a virtualized SW architecture by means of different hypervisor technologies. The main goal is to evaluate several characterization metrics (i.e., timing, mutual exclusive shared memory access overhead, and SW memory footprint) by also defining a methodology to compare different virtualization technologies in specific domains while considering the avionic/space one as a driving example. The main contributions of the paper are related to:

1) the definition of a characterization approach able to identify and classify HPV in terms of main services offered by the virtualized environment;

2) the exploitation of specific HPV technologies in space domain (i.e., Xtratum and PikeOS), running on different multi-core HW configurations;
3) the identification of an extensible set of benchmark functions useful for HPV characterization.;

The remainder of the paper is organized as follows: Section II presents related works that treat the problem of benchmarking analysis (with main focus on timing performance analysis) in different domains. Section III proposes a benchmarking analysis methodology defined to evaluate and compare HPV characterization metrics. Section IV applies the methodology by means of several experiments performed to compare two reference HPVs (i.e., *Xtratum* and *PikeOS*) on different multi-core platforms (i.e., quad-LEON3 on Virtex-6 FPGA [16], GR740-LEON4 [17]) recently adopted in the space domain. Finally, Section V closes the paper with some conclusions and future works.

II. Background and Related Works

Since space-grade processors have been well analyzed in literature, with specific hardware [18] and bare-metal benchmarking analysis [19–24], space industries have decided to invest greater efforts in SW validation and qualification w.r.t. HPV-based solutions. In fact, as already done in the cloud/server domain, where HPVs are the most used SW isolation tools, different works have been done in the last years, with focus on HPV benchmarking analysis, also in the embedded domain (HPVs are becoming very important for their isolation features, but the main problem is related to certification issues).

For a better understanding of the topic, it is firstly necessary to introduce the well-know different HPV types: the so called Type-1 and Type-2. The former considers an operating system running on a virtual machine (VM) in user mode that executes a privileged instruction. In this scenario, the hardware trap to the hypervisor so that the instruction can be emulated in software. The latter is composed by user programs that interpret the machine's instruction set, which also creates a VM. The OS running on the hardware is called the Host OS, while the OS running on the VM is called Guest OS. Type-2 HVPs translate the binary programs of the guest operating system block by block, replacing certain control instructions with HPV calls. The translated blocks are cached and executed directly to improve performance. Both Type-1 and Type-2 HPVs work with unmodified guest operating systems, but have to jump through hoops to get reasonable performance. A different approach that is becoming popular (also on the embedded domain) is to modify the source code of the guest operating system so that, instead of executing sensitive instructions at all, it makes hypervisor calls. In such a scenario, the HPV must define an interface consisting of a set of hypervisor calls that a guest OS can use. This set of calls forms what is effectively an API (Application Programming Interface) even though it is an interface for use by guest OS, not application programs. A guest OS from which sensitive instructions have been intentionally removed is said to be "paravirtualized". Thus, the following paragraphs briefly analyze some of the works in cloud/server and embedded domains.

In the work in [25], the HPV performance tests were conducted on a single physical machine with an *Intel Core2*

Duo E8400 dual-core (clock speed of 3GHz) equipped with 4GB of DDR2 RAM. Each HPV (i.e., *Hyper-V* [26], *ESXi* [27], *OVM* [28], *Xen* [29]) was tested in an isolated manner, with a complete hard drive format between installations. An *Ubuntu 12.04 LTS* virtual machine was created inside each considered HPV partitions. Then, standard benchmark suites to compare performance metrics of main hardware components, i.e., CPU (*nbench*), NIC (*netperf*), storage (*Filebench*), memory (*ramspeed*), have been used.

The paper in [30] benchmarks the performance metrics of the different HPV considering different parameters related to CPU (execution time performances), RAM, DISK *Read-Write*, *Network Read-Write*. The setup involved three HPVs, namely *Virtual Box 6.0* [31], *VMware workstation* [32] as a hosted HPV and *Xen* as native or Type-1 HPV. *Glances* [33], a free tool to monitor Linux OS from a text interface, has been used to collect and monitor the traces. HW and SW requirements for the experiment of hosted HPV consisted of a PC equipped with an i5-4590 @3.30 GHz processor, 4GB DDR2 RAM, 500GB of hard drive with Windows 7 OS. The HW and SW requirements of native HPV consisted of a bare metal PC equipped with i5-4590 @3.30 GHz processor, 4GB DDR2 RAM, 500GB with *Virtual Machines* (VMs) created by means of *Xen Center* [29]. Each VM has the following configuration: i5-4590 @3.30 GHz processor, 2 GB DDR2 RAM and 40GB hard drive, the *Guest OS* on the VM is Ubuntu Linux 14.04 LTS. From the experiments carried out, they has concluded that best CPU performance is given by Xenserver 6.5 [29], while best RAM performance is given by VMware.

In the context of embedded systems, Shuja et al. [34] presents a survey on software- and hardware-based mobile virtualization techniques, considering challenges and issues faced in virtualization of CPU, memory, I/O, interrupt, and network interfaces. In this survey, the authors list a lot of HPV technologies and propose a classification of such kind of virtualized environments respect to virtualization type (i.e., type-1 and type-2), hardware platforms (i.e., ARM, Intel Atom, MIPS, and PowerPC), CPU virtualization model (i.e., binary translation, para-virtualization, and HW assisted), memory protection model (i.e., fixed partitions, domain access control tagging, and VMID tagging), Interrupt virtualization model, I/O access model (i.e., dedicated, or shared), and network virtualization model. This work does not apply specific benchmark activities, but defines formal requirements of virtualization that are the same in several different domain (e.g., avionics and space).

Finally, considering space domain, [35] proposes a real-time hypervisor for space applications assisted by COTS hardware. The system has been deployed and evaluated on a Xilinx ZC702 (Zynq-based) board, exploiting the ARM TrustZone architecture. The authors focus on memory footprint, context switch overheads and real-time performance of the RTOS services (i.e., cooperative scheduling, preemptive scheduling, interrupt processing, interrupt preemption processing, synchronization processing, message processing, and memory allocation). The work lacks in the evaluation steps since it considers only single-core configuration, while disabling MMU, caches and branch prediction support for guest OS.

The work in [36] illustrates the results related to the use of the quad-core GR-CPCIXC4V LEON4 RASTA

applications, in SMP configuration, jointly with the use of the PikeOS HPV. They consider the *Dhrystone* benchmark [37] with enabled compiler optimizations and float point emulation. The tests have been done in three different cases, bare metal, PikeOS on single core and PikeOS on multicore. They even tested the scheduler precision, calculating the difference between the real-time activation of a partition and the theoretical one. In this paper a deep analysis of the HW/SW architecture has been presented, but scheduler analysis is not properly accurate due to the lack of tests with the optimization suggested by SYSGO [38].

Table 1 Related Works Classification

Work	Target Architecture	Domain	HPV	Benchmarks
[25]	Intel Core 2 Duo E8400	PC/SERVER	Hyper-V, ESXi, OVM, VirtualBox, Xen	CPU test (nbench), NIC Test (Netperf), HDD Test (Filebench), Memory test (ramspeed)
[30]	Intel i5-4590	PC/SERVER	Vmware, Xen	Glances
[34]	ARM, PowerPC, MIPS, Intel Atom	EMBEDDED	Several HPV	-
[35]	ARM Cortex-A9	EMBEDDED	RTZVisor	Thread-Metric Benchmark
[36]	GR-CPCIXC4V LEON4 RASTA	EMBEDDED	PikeOS	Dhrystone
[39]	NGMP space	EMBEDDED	Xtratum	Drystone, CoreMark
[40–42]	GR-CPCI-LEON4-N2X	EMBEDDED	Xtratum, PikeOS	Ad-Hoc benchmark

The work in [39] proposes the porting of *Xtratum* HPV [43] on the ESA NGMP prototype, i.e., a quad-core 32-bit LEON4 (compatible with *Sparc V8 Instruction Set Architecture*) running at 50 MhHz, with 4x4Kb instruction and data Level-1 caches, a shared 256 KB Level-2 cache, MMU, IOMMU and two shared FPUs. Then, a performance testing has been made to consider three different implementation aspects: the HPV layer, the *Partition Context Switch* (PCS) and the influence of the multi-core shared HW resources (memory and FPU) management overhead introduced on the execution. Two well-known standard benchmarks have been used: Dhrystone [37] and CoreMark [44]. These benchmarks have been run both as bare-metal applications and as partitions running on top of XtratuM. The partitioned benchmarks have been executed under the hypervisor cyclic scheduling, with a time slot larger than 30 seconds for the dhrystone executions, to avoid partition context switch, and different execution slot durations for Coremark, to evaluate partition context switch impact. Results show very low performance loss in the case of the Dhrystone benchmark and a partition context switch overhead in the range of 149 to 151 microseconds.

The works proposed in [40–42], arising from the ARTEMIS-JU AIPP 2013 621429 EMC2 (*Embedded Multi-Core systems for Mixed Criticality applications in dynamic and changeable real-time environments*) European project [45–48], provides a study of the PikeOS 3.5 and Xtratum 4.2.1 Architectures, with the exhaustive description of a specific use case: the communication between partitions with the utilization of the shared memory, queuing ports, and sampling

ports. The HW used for this test is a Leon 4 with the Cobham Gaisler GR-CPCI-GR740 board [17]. Xtratum provided a better performance in both cases. The limitation of this work is related to the fixed HW platform and the fixed SW partition scenario, but this is a very good starting point to evaluate HPV features. In fact, the work presented in this paper extends the previous ones ([40–42]) analyzing the possible HPV behavior on different multi-core architectures, while trying to characterize different HPVs (i.e., Xtratum and PikeOS) using an extensible approach. Table 1 summarizes the considered works, classifying them with respect to boards, HPV and benchmarks.

III. Benchmarking Analysis Methodology

Starting from the related works, the goal of this Section is to define a methodology for benchmarking analysis of HPV features considering characterization metrics in the embedded domain. In this case, as highlighted before, safety requirements are mainly the ones related to "isolation" relevant in the space domain and related to several standard directives (i.e., *European Cooperation for Space Standardization*, ECSS, DO-297, ARINC-653, NPD, NPR, OST [49]). Virtualization aims at providing time and space isolation [50] among the existing resources in the same hardware platform [51]. The ongoing trend goes towards multi-core technology. However, multi-core technology brings several challenges specially at the micro-architectural level (i.e., shared caches [52], shared buses [53]). In this context, this work investigates multi-core processor architectures that exploit virtualized environments by means of different HPV technologies, with emphasis on characterization.

The proposed methodology is composed of different activities, as shown in Fig. 1. The first one is related to the selection of the benchmark to be used. It is typically composed of functions taken from commercial or academic domain [37, 44, 54–56]. Such functions should be able to "stimulate" all the features of interest with respect to the system under analysis (e.g., timing, memory access, scheduling, communication, etc.). The benchmark composition is strictly related to the second activity, i.e., the selection of the characterization metrics of interest (e.g., DMIPS, shared memory access overhead, memory footprint, etc.). Finally, since the functions execution is affected by the selected compiler optimizations (e.g., O0, O1, O2, O3 for GCC) and the actual HW/SW architecture (i.e., number of cores, memory architecture, OSs, HPVs), the third activity focuses on the definition of such architectural parameters.

Then, the obtained results will drive developers to check if the selected technologies are able to fulfill Non-Functional requirements and eventually make a comparison in order to adopt the best suitable one. The remain of this section presents with more detail the benchmark functions, the reference HW (i.e., multi-core) and SW (i.e., HPV) architectures used to perform the experimental benchmark analysis.

A. Benchmark Functions and Characterization Metrics

The benchmark has been selected to address three different characterization metrics: HPV overhead, shared memory access overhead, and SW memory footprint. So, to evaluate characterization metrics in different HPV scenarios, this

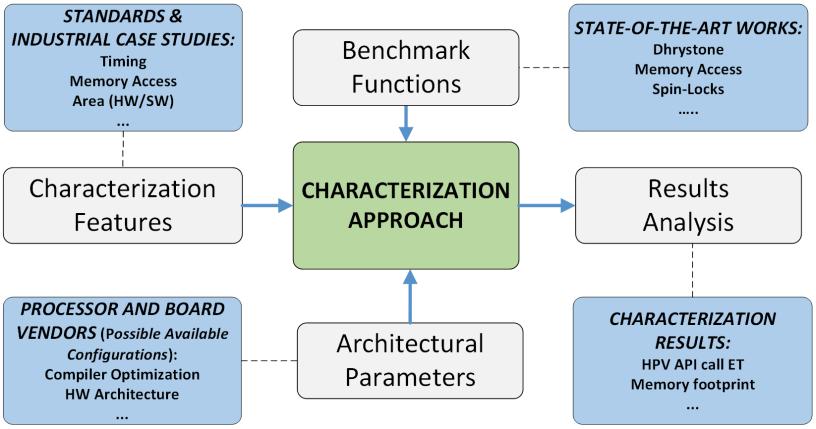


Fig. 1 Benchmarking Methodology.

work exploits the following two functions: the well-known *Dhrystone* benchmark for timing characterization (DMIPS, *Dhrystone Million Instructions Per Second*), *spin-lock stress tests* for shared memory access overhead. Both the functions are then considered to evaluate SW memory footprint in terms of KB.

The Dhrystone function is a synthetic benchmark working on integer operations. Therefore, the operations evaluates the basic CPU load without an intensive use of the stack [39]. This function has been compiled with different optimization flags, and has been executed on all the reference boards (see below). It is worth noting that two different versions of the *Dhrystone* have been adopted. The first version relies on the different HPV's API, the second one exploits the functions provided by the standard C++ library (without re-defining and re-implementing standard operations).

The *spin-lock stress test* function stresses the shared memory with a simultaneous access from all the HPV partitions. This is done by running one thread in each partition, assigning each partition to a single core, while increasing the number of partitions from 1 to 62 in order to evaluate the memory access overhead taking onto account intra-core and inter-core scheduling. In this function, the memory access has been managed by means of two kind of semaphores, the *Binary semaphore* and the *Semaphore with HPV API*. The former can assume two values, 0 or 1. In the first case the memory is free and the thread can access data, otherwise, the memory is busy and the thread must wait. In such a scenario no queues have been implemented, so it is possible to directly evaluate CPU spinning effect. The latter exploits HPV API for thread management. Therefore, if a thread is waiting, until the memory is busy, it is possible to lock it in kernel space instead of stressing the system with continuous spinning. This approach can be found in other domains like [57], where an analysis on thread scheduling has been performed, with focus on CPU's wasted cycles on spinning and the total number of CPU cycles (total execution time). The function, that run in each user partition, checks if the memory is free, then locks the memory and performs read and write operations in the main memory. If the memory is not free, the thread waits until the memory becomes free. This operation is repeated 10.000 times for each thread, as shown in Fig. 2 . In this way it is possible to evaluate thread's contention over semaphores since the tests make an

extensive use of the system bus.

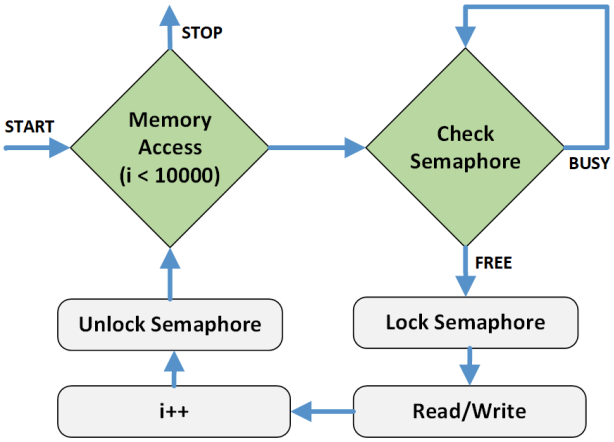


Fig. 2 Spin-Lock Memory Stess Test Approach.

B. Reference HW (Multi-core) Architectural Parameters

In this work, the GAISLER LEON3 [16] and LEON4 [58] *General Purpose Processors* (GPP), mounted on three different boards, have been considered.

LEON3 is a 32-bit synthesizable soft-processor that is compatible with SPARC V8 architecture: it has a seven-stage pipeline and Harvard architecture, uses separate instruction and data caches and supports multiprocessor configurations in Symmetric Multi-processor (SMP) mode. LEON4 is a synthesizable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture and supports MUL, MAC and DIV instructions, IEEE-754 floating-point unit (FPU) and Memory Management Unit (MMU). It has also a separate I/D multi-set Level-1 (L1) caches, an optional Level-2 (L2) cache, with snooping, a 7-stage pipeline, with branch prediction, and an AMBA-2.0 AHB bus interface. These two processors actually represent the reference processors for space applications.

Table 2 LEON3 Multi-Core Area Occupation (with different Cache Configurations)

Logic elements	2/4/32/LRU*	2/1/32/LRU*	4/4/32/LRU*	2/1/32/Random*	4/4/32/Random*
Slice Registers	37145 (12%)	36927 (12%)	39904 (13%)	36772 (12%)	37309 (12%)
Flip Flops	37135	36917	39894	36762	37299
Slice LUTs	89914 (59%)	89489 (59%)	92029 (61%)	89428 (59%)	90434 (60%)
Logic LUTs	88431 (58%)	87804 (58%)	90590 (60%)	87625 (58%)	88889 (58%)
Memory LUTs	1233 (2%)	1425 (2%)	1233 (2%)	1425 (2%)	1233 (2%)
Occupied Slides	30226 (80%)	29838 (79%)	31262 (82%)	27225 (72%)	29797 (79%)
LUT Flip FLop pairs	94734	94362	97909	93350	95230

* Way/Set Size/Bytes per Line/Replacement Policy

The quad-core LEON3 FPGA reference architecture is shown in Fig. 3, while Table 2 shows the different area

occupation. The considered development board is the Xilinx Virtex-6 FPGA [59] with 512 MB ram (with 301440 slice registers, 150720 logic LUTs, 58400 memory LUTs, 37680 occupied slices). Starting from GRLIB, a VHDL library of IP cores source codes for designing a complete system-on-chip, LEON3 processor has been customized with a system clock of 75 MHz per core and the following characteristics:

- from 1 to 4 Cobham Gaisler LEON3 SPARC V8 Processor connected with AHB shared bus in SMP configuration;
- 8 register windows;
- 2 way * 8 KiB set size for instruction set-associative cache, with 32 bytes per line with Least-Recently-Used (LRU) replacement algorithm and write-through policy;
- 5 set-associative data cache configurations with write-through policy:
    1) 2 way * 4 KiB cache set size, with 32 bytes per line with LRU replacement algorithm (Default);
    2) 2 way * 1 KiB cache set size, with 32 bytes per line with LRU replacement algorithm;
    3) 4 way * 4 KiB cache set size, with 32 bytes per line with LRU replacement algorithm;
    4) 2 way * 1 KiB cache set size, with 32 bytes per line with Random replacement algorithm;
    5) 4 way * 4 KiB cache set size, with 32 bytes per line with Random replacement algorithm;

.

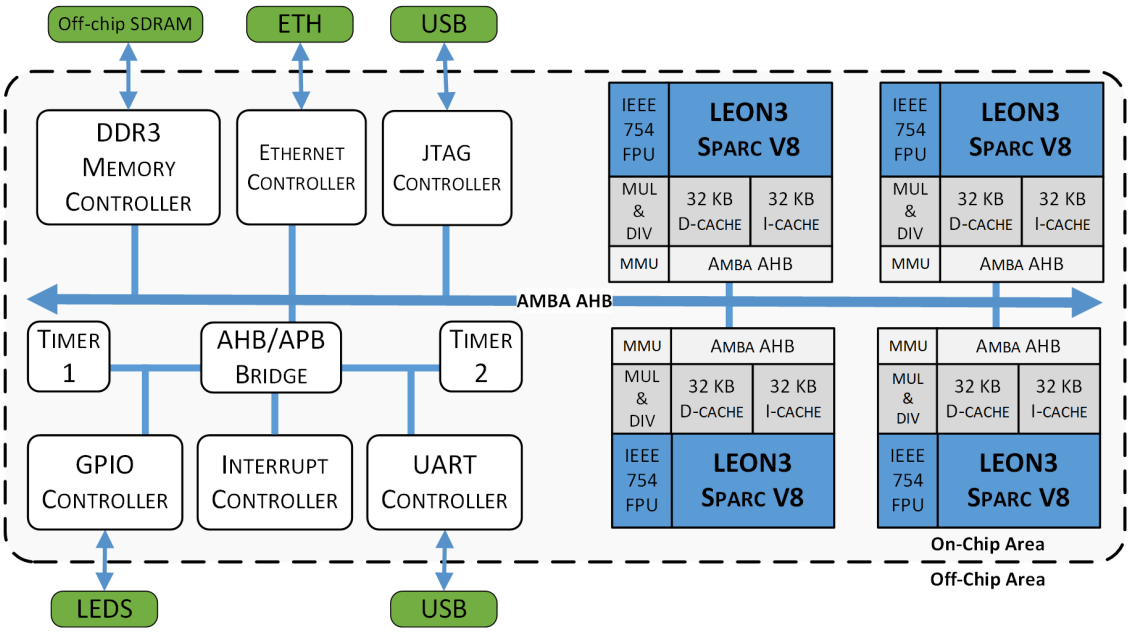


Fig. 3 Quad-core LEON3 Hardware Architecture.

The LEON4 architectures is related to the GR740, a radiation-hard system-on-chip with a quad-core fault-tolerant LEON4 SPARC V8 processor, with SpaceWire router, PCI and CAN 2.0 interfaces. The GR740 has been designed as the European Space Agency's Next Generation Microprocessor (NGMP) and is part of the ESA roadmap for standard

microprocessor components.

C. Reference SW (HPV) Architectures

In order to analyze different virtualization solutions in the space domain, the reference HPVs chosen in this work are PikeOS [12], distributed by SysGO, and Xtratum, offered by Fentiss [43].

1. PikeOS Hypervisor

PikeOS, developed by SysGO company [38], is a multi-architecture real-time operating system able to provide paravirtualization services with support for different architectures (i.e., PowerPc, x86, SPARC V8, ARM, MIPS). As represented in Fig. 4, the PikeOS architecture is structured in four layers, divided into user and kernel mode. The *User Mode* is composed of a *Partitions layer*, where the user partition applications run, and a *PikeOS System Software* (PSSW), i.e. a kernel module providing the partition abstraction. The PSSW is responsible for the creation of partitions and threads at boot time. The system configuration is stored in a dedicated database, the *Virtual Machine Initialization Table* (VMIT), specifying the partitions, their resources and access and control rights. The *Kernel Mode* consists of a *PikeOS Separation Microkernel*, a separation kernel supporting real time performance, and a *Architecture and Platform support package*, that encapsulates all the details regarding the CPU architecture and the overall hardware platform. PikeOS is based on a microkernel architecture directly inspired by the L4 kernel specification and support tasks (virtual address spaces), threads (execution entities), thread scheduling and inter-thread communication. PikeOS is not usually considered as a hypervisor, in fact it is a separation kernel. However, it has been used in the hypervisor mode of that kernel in this work. A Memory Management Unit is required in order to support a separate memory space for each task. The PikeOS software architecture is shown in Fig. 4. The reference PikeOS version is the 4.2.3668 released in 2019.

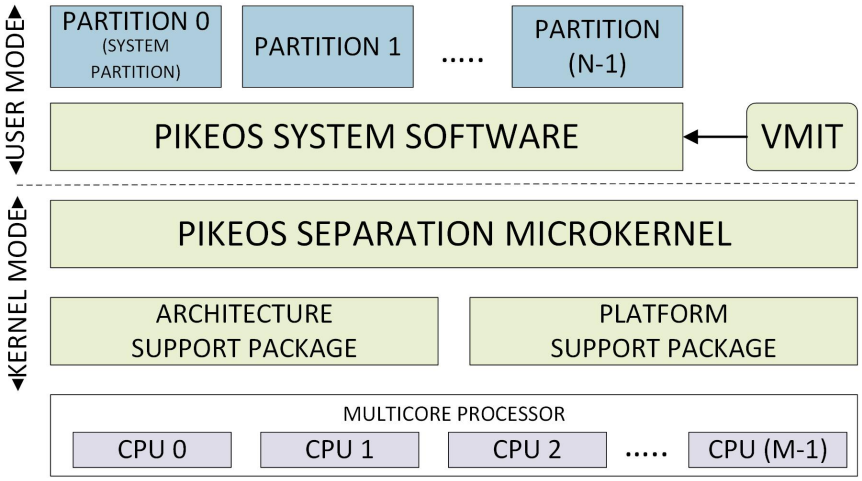


Fig. 4 Hypervisors Under Investigation - PikeOS Software Architecture.

2. Xtratum

XtratuM is a type-1 hypervisor, developed by Fentiss SL [60], a spin-off from *Universidad Politecnica de Valencia*, supporting paravirtualization for multiple architectures. XtratuM natively supports SPARC architecture and LEON processors. Fig. 5 shows the software architecture of the XtratuM hypervisor, which includes a *Partitions layer*, divided in *Supervisor Partitions* (able to handle Xtratum health monitors features), *User Application Partitions* (i.e. the isolated execution environments), *XtratuM Hypercall Interface*, i.e. the set of hypercalls used to access the paravirtualized services supported by Xtratum, and *XtratuM kernel*, a monolithic, non-preemptable kernel executed in the supervisor mode of the target processor, supporting hardware platform virtualization (i.e. CPU, memory, interrupts, and critical peripherals), partition scheduling, inter-partition communication and health monitoring features. Partition code has to be paravirtualized in order to run on top of the hypervisor and Xtratum do not support any native form of concurrency inside a partition (i.e. it is needed to an operating system inside partition to manage parallel tasks in isolated mode, with the use of vcpu instances).The reference Xtratum version is the 4.5.0 released in 2019. Xtratum has been considered in multi-core scenario for LEON4, and in single-core scenario for the LEON3 architecture since these was the only licenses offered by Fentiss for this work.

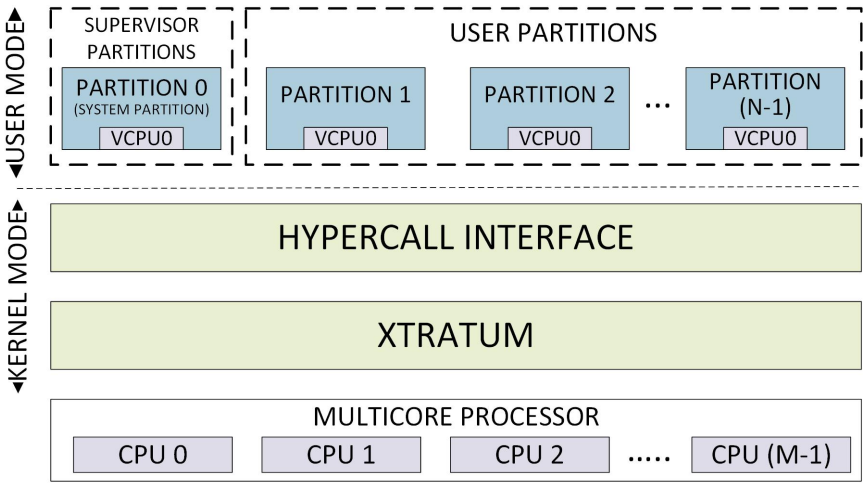


Fig. 5 Hypervisors Under Investigation - Xtratum Software Architecture.

IV. Experiments, Results, and Analysis

This section presents the experimental results obtained by applying the benchmarking analysis methodology described below to characterize HPVs features.

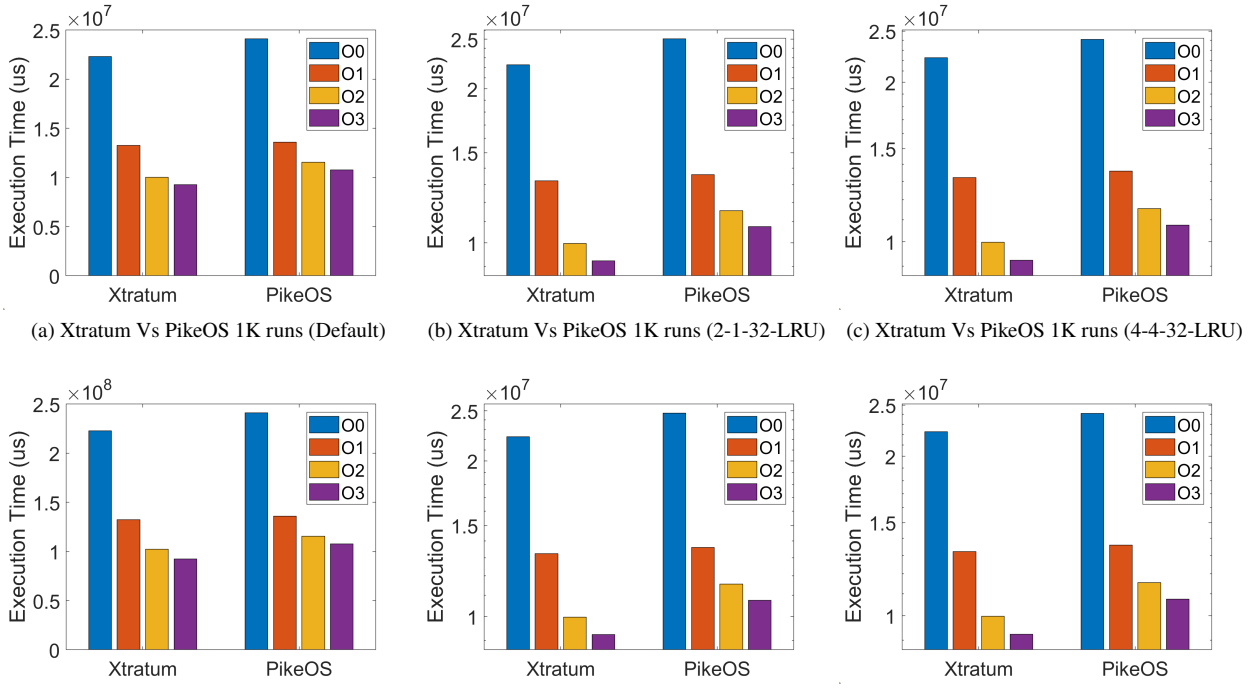
A. Timing Behavioral

The timing characterization analysis involves the use of Dhrystone benchmark to compare HPV characterization metrics. The bare-metal Dhrystone results are 18.7 s for O0, 9.7 s for O1, 6.1 s for O2 and 5.4 s for O3 considering

1.000.000 runs on LEON3 ML605 platform. The bare metal Dhrystone results for GR740 can be found on Gaisler website [61]. The adoption of HPVs into a multi-core scenario offers the opportunity to exploit partitions allocated on different cores. Therefore, the main idea is to replicate Dhrystone in different partitions, each one allocated on a different core, so it is possible to evaluate multi-core isolated HPV behavior.

1. ML605 LEON3 Scenario

Fig. 6 presents different Dhrystone tests made with PikeOS and Xtratum on the LEON3 reference HW architecture by first considering a single-core scenario. From such a plot it is possible to note that the number of Dhrystone runs do not change the execution time and the related timing behavior (the execution time is scaled linearly with the number of runs), so the test is repeatable and it is possible to fix once for all the number of runs. With respect to the Dhrystone execution time, Xtratum generally performs better than PikeOS (especially when considering cache and optimization flags).



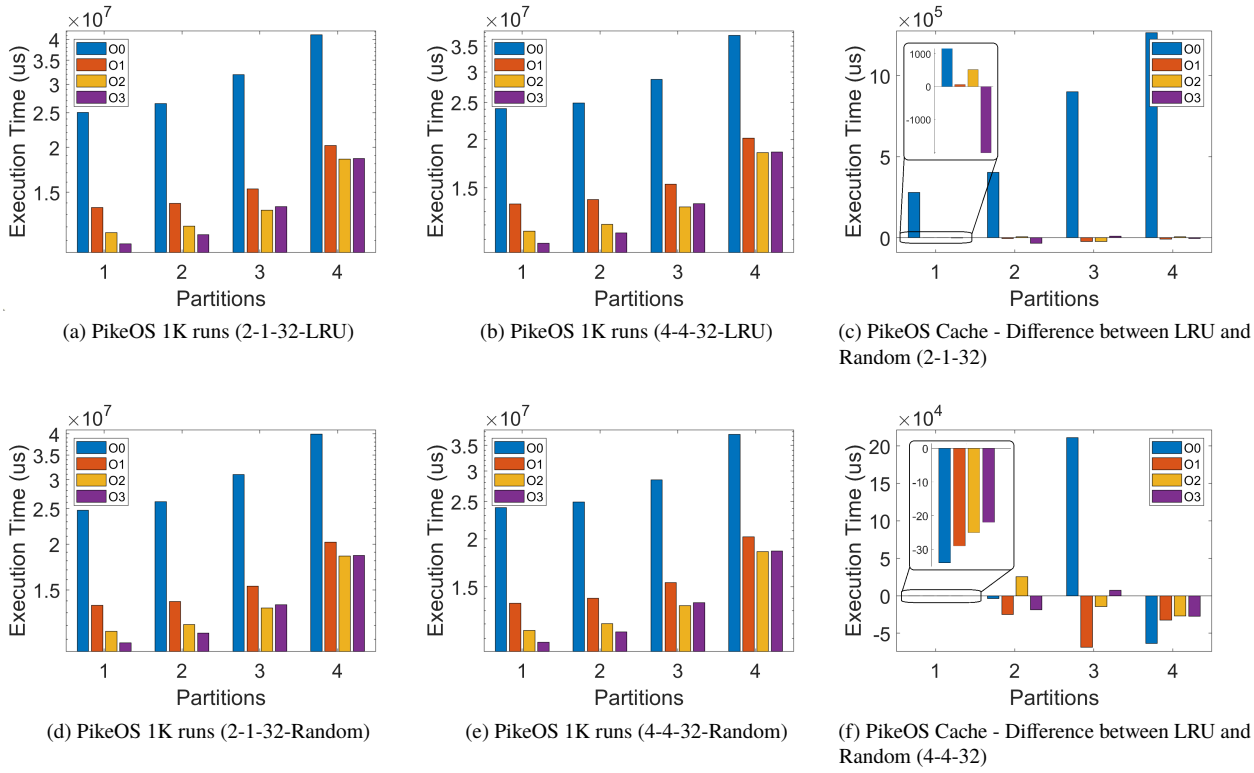
(a) Xtratum Vs PikeOS 1K runs (Default) (b) Xtratum Vs PikeOS 1K runs (2-1-32-LRU) (c) Xtratum Vs PikeOS 1K runs (4-4-32-LRU)

(d) Xtratum Vs PikeOS 10K runs (Default) (e) Xtratum Vs PikeOS 1K runs (2-1-32-Random) (f) Xtratum Vs PikeOS 1K runs (4-4-32-Random)

Fig. 6 Xtratum Vs PikeOS LEON3 Single-core.

Considering the multi-core scenario, since the only currently available HPV supporting multi-core scenario is the PikeOS one, Fig. 7 presents the different timing characterization metric obtained by considering replicated versions of Dhrystone executing in different partitions (from 1 to 4) allocated on different cores. It is worth noting that the increased number of cores introduces some negative effects, since bus contention and HPV overhead can negatively affect the timing characterization metric (i.e., for 3 and 4 partitions/cores the O3 optimization increases the benchmark execution

time). As shown in Fig. 7 (c) and (f), the effects of cache size replacement policy changes drastically with the cache data size, while it is possible to have some anomalies in the allocation environment because the different configurations can affect the behavior in different manner. Further analysis will be made in future to analyze cache configuration and find the best data cache size, considering also other benchmark functions.



(a) PikeOS 1K runs (2-1-32-LRU)

(b) PikeOS 1K runs (4-4-32-LRU)

(c) PikeOS Cache - Difference between LRU and Random (2-1-32)

(d) PikeOS 1K runs (2-1-32-Random)

(e) PikeOS 1K runs (4-4-32-Random)

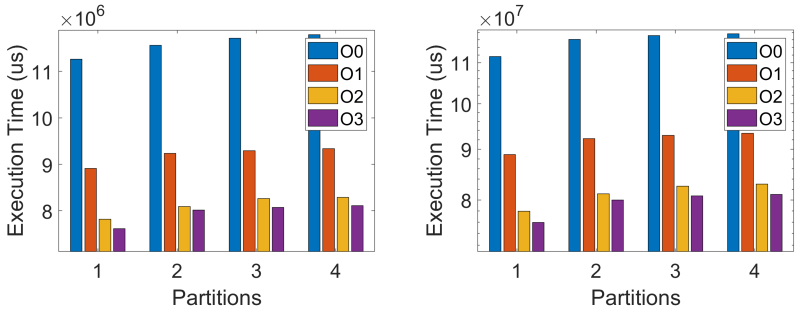
(f) PikeOS Cache - Difference between LRU and Random (4-4-32)

Fig. 7 PikeOS LEON3 Multi-core with different cache configurations (1 partition per core).

2. LEON4 GR740 Scenario

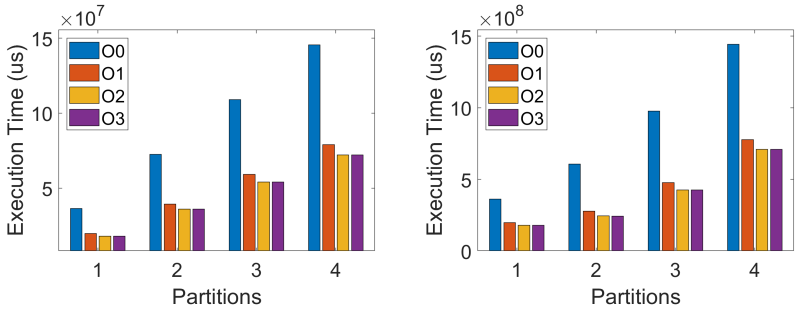
The GR740-LEON4 scenario has a behavior similar to the LEON3 architecture in terms of Dhrystone execution time. The number of runs do not affect the behavior, and there is a decreasing histogram trend. It is worth noting that Xtratum has an improvement also considering the O3 flag, while PikeOS has no gain in timing execution among O2 and O3 optimization flags, as shown in Fig. 8 and Fig. 9.

Xtratum offers the possibility to decrease execution time, increasing timing characterization metrics, by using a specific optimization flag offered by the reference toolchain [62]. This optimization option increases timing metric in terms of about 13%, while offers the same repeatable behavior of the standard compilation flow, as shown in Fig. 10.



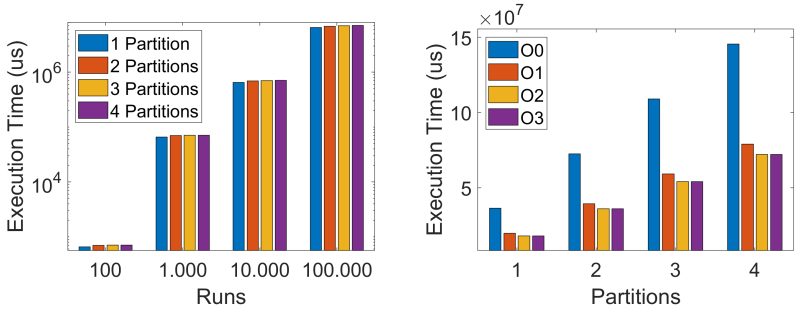
(a) Xtratum 1.000.000 (b) Xtratum 10.000.000

Fig. 8 Dhrystone Execution on Xtratum running on GR740 LEON4 Multi-Core Architecture (No Compiler Optimization, 1 partition per core).



(a) PikeOS 1.000.000 (b) PikeOS 10.000.000

Fig. 9 Dhrystone Execution on PikeOS running on GR740 LEON4 Multi-Core Architecture (1 partition per core).

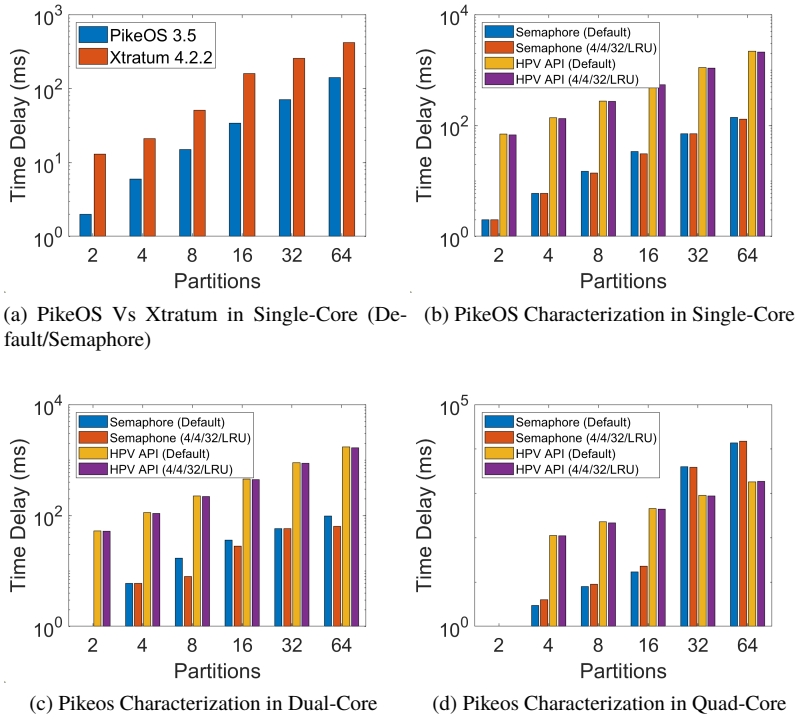


(a) Xtratum V8 Optimization Trend (repeatability test) (b) Xtratum optimization improvements (1.000.000 Dhrystone Runs)

Fig. 10 Analysis of Dhrystone Execution on GR740 LEON4 Multi-Core Architecture using Xtratum optimization flags (1 partition per core).

B. Memory Access Spin-lock Stress Test

The activities to evaluate the second characterization metric, as described in Section III.A, are based on the LEON3 FPGA scenario in order to evaluate cache interference. It is worth noting that LEON4 has not been considered since it is not possible to change cache configurations. Xtratum and PikeOS are compared only in single core, since the current available version of Xtratum supports only LEON3 single-core configuration. PikeOS has been characterized also in multi-core scenario. The LEON4 board will be considered for future works, also considering the possibility to run real-time operating systems on the top of each software partitions (i.e., RTEMS [9]). Fig. 11 presents some results related to the spin-lock stress test.



(a) PikeOS Vs Xtratum in Single-Core (Default/Semaphore)

(b) PikeOS Characterization in Single-Core

(c) Pikeos Characterization in Dual-Core

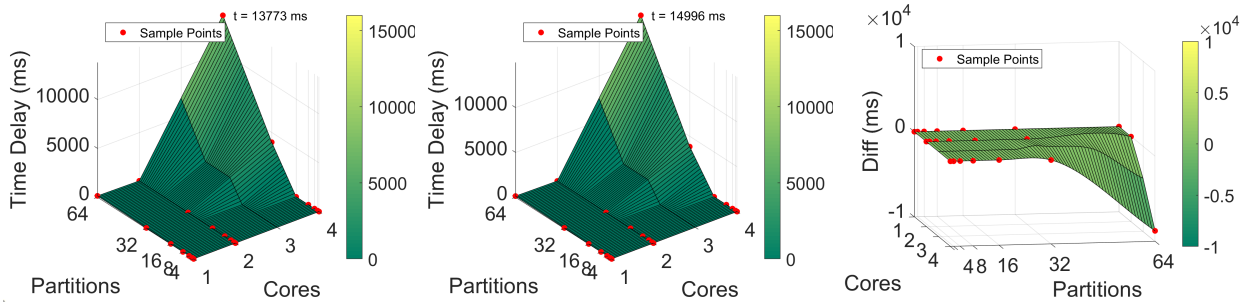
(d) Pikeos Characterization in Quad-Core

Fig. 11 HPV characterization on LEON3 Multi-Core Architecture.

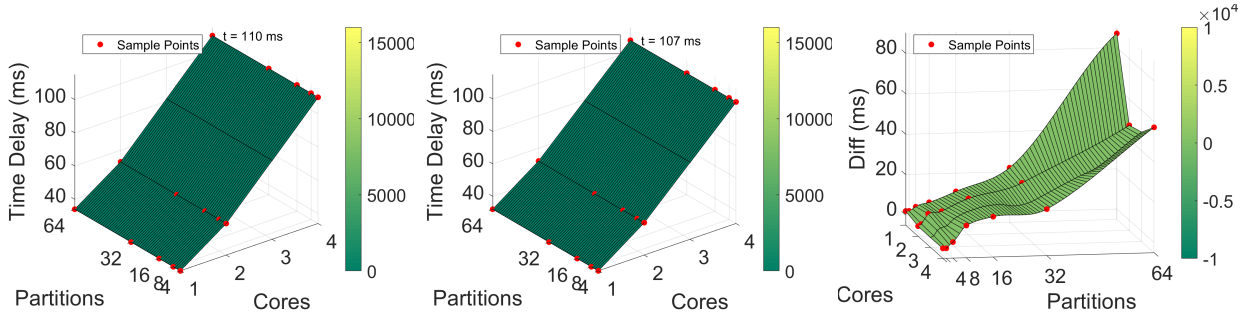
Xtratum uses the standard POSIX C semaphores implementation, while PikeOS offers the possibility to use standard semaphores or custom HPV *Application Programming Interface* (API) to handle the accesses to shared memory. In the single core scenario, Xtratum shows better characterization metrics w.r.t. semaphore delays, also considering the growth of HPV SW partitions from 2 to 64, as shown in Fig. 11 (a). Considering multi-core scenario, PikeOS has better behavior in terms of semaphore handling when using the HPV API offered by *Sysgo*. In fact, in the 4-core scenario with 32 and 64 partitions the use of PikeOS primitive guarantees a fixed and predictable behavior, while the standard POSIX implementation introduces overheads into the semaphores management, as shown in Fig. 11 (d). Such a result is an indirect information related to bus and cache behaviors. The increased data cache size seems to show worse execution

times, so the cache configuration can affect in a bad manner system execution time, as shown in Fig. 11 (b) and (c).

By focusing on cache interference, Fig. 12 shows some interesting results related to different cache configurations and semaphore implementations in a single view. By considering at the same time number of cores and number of partitions, Fig. 12 (a) and (b) show that the overhead related to semaphore handling increase with the number of cores and partitions per cores when using standard POSIX implementation. Fig. 12 (d) and (e) show that the overhead related to the semaphore handling is fixed when using HPV API implementations, so it changes only with the number of cores. Fig. 12 (c) and (f) show that the increased data cache size decreases space in terms of about 1% w.r.t. the number of partition per cores. Using PikeOS primitive can increase a little bit the execution time w.r.t. the number of partitions, while it decreases the execution time in terms of less then 1% respect to the single-core scenario.



(a) PikeOS Semaphore Overhead (Default) (b) PikeOS Semaphore - Overhead (4/4/32/LRU) (c) PikeOS Semaphore - Cache Inference



(d) PikeOS API Overhead (Default) (e) PikeOS API - Overhead (4/4/32/LRU) (f) PikeOS API - Cache Inference

Fig. 12 PikeOS Characterization on LEON3 Multi-Core Architecture.

C. SW Memory Footprint

Finally, to take a look also to the memory footprint, Table 3 illustrate the size associated to the different binary applications related to Dhrystone. The optimization flags do not changes PikeOS size, while changes the Xtratum one, since the compiling toolchain tries to optimize also memory footprint by working on partitions allocation, while PikeOS only instantiate partition at compiler time.

Table 3 Memory Footprint for different Dhrystone application scenarios For GR740 (O3 Optimization flags).

Configuration	PikeOS GR740 (bytes)	Xtratum GR740 (bytes)	PikeOS LEON3	Xtratum LEON3
1-core	709324	168532	608284	113088
2-core	709324	180680	608284	-
3-core	709324	192964	608284	-
4-core	709324	205579	608284	-

V. Conclusion

This work has presented a specific methodology to characterize HPVs by means of benchmark functions.The results show that Xtratum behaves better then PikeOS in timing and memory access scenarios, while PikeOS offers hypervisor features that seems to be more predictable in introduced overhead. Future works will considers other classical benchmark functions (e.g., Whetstone [63], CoreMark [44], SPEC [54]), and domain-oriented industrial benchmarks able to test and stress the system in a systematic way to extract information about system behavior and overhead (i.e., interrupt handling, context switch overhead, etc.).

Appendix

Table 4 Xtratum Vs PikeOS LEON3 Single-core (us)

HPV	Opt.	Default (1K)	Default (10K)	2-1-32-LRU (1K)	2-1-32-Random (1K)	4-4-32-LRU (1K)	4-4-32-Random (1K)
Xtratum	O0	22305813	222711487	22267547	22267554	22267373	22267387
	O1	13265615	132309209	13227205	13232673	13227143	13227160
	O2	10012106	102309210	9973722	9977139	9973679	9973680
	O3	9265448	92307631	9227044	9231548	9227016	9227015
PikeOS	O0	24107573	222711487	25033024	24754021	24099737	24099771
	O1	13600669	132309209	13601462	13601400	13597248	13597277
	O2	11557541	102309210	11559736	11559222	11556437	11556462
	O3	10757414	92307631	10758513	10760527	10756561	10756583

Table 5 PikeOS LEON3 Multi-core with different cache configurations, 1 partition per core (us)

		LRU				Random			
Cache size	Partitions	O0	O1	O2	O3	O0	O1	O2	O3
2-1-32	1	25033024	13601462	11559736	10758513	24754021	13601400	11559222	10760527
	2	26502692	13949365	12070309	11405991	26100026	13955080	12064589	11439668
	3	31881737	15318571	13372407	13668971	30981693	15341872	13395315	13659683
	4	41213561	20246261	18549853	18611293	39949152	20255872	18545084	18617484
2-1-32	1	24099737	13597248	11556437	10756561	24099771	13597277	11556462	10756583
	2	24912486	13966271	12047909	11433987	24916003	13991378	12022655	11452728
	3	28703203	15321285	13377071	13637957	28492102	15390122	13391291	13630644
	4	37296614	20191790	18499209	18569179	37360350	20224441	18526303	18596690

Table 6 Dhrystone Execution on Xtratum running on GR740 LEON4 Multi-Core Architecture, 1 partition per core (us)

		1K Runs				10K Runs			
HPV	**Partitions**	**O0**	**O1**	**O2**	**O3**	**O0**	**O1**	**O2**	**O3**
Xtratum	1	11264820	8912517	7820316	7616269	111600995	88880651	77920573	75920567
	2	11562395	9232951	8090530	8017215	116105796	92255951	81157437	79977384
	3	11715902	9295563	8266172	8073142	117145109	92948159	82578158	80773440
	3	11788596	9338213	8288445	8112871	117652794	93394673	82990271	81006941
PikeOS	1	36342850	19719728	18029314	18029203	363442200	197209200	180296000	180291600
	2	72575142	39325650	35945800	35944041	606776000	277838800	244728800	243172600
	3	108950889	59057070	53980056	53979456	975918600	476901200	426195400	426049600
	4	145419140	78881239	72122283	72121979	1442384200	777361200	709720400	709392000

Table 7 HPV characterization on LEON3 Multi-Core Architecture. (ms)

				Partitions					
HPV	**Cores**	**Technique**	**Cache Size**	**2**	**4**	**8**	**16**	**32**	**64**
PikeOS	1	Semaphore	Default	2	6	15	34	71	141
		Semaphore	4-4-32	2	6	14	31	71	130
		API	Default	70	138	278	556	1109	2192
		API	4-4-32	68	134	272	545	1087	2105
Xtratum	1	Semaphore	Default	13	21	51	160	257	418
PikeOS	2	Semaphore	Default	1	6	17	36	58	98
		Semaphore	4-4-32	1	6	8	28	58	64
		API	Default	53	113	227	456	898	1725
		API	4-4-32	52	109	220	443	878	1678
PikeOS	4	Semaphore	Default	-	3	8	17	4022	13829
		Semaphore	4-4-32	-	4	9	23	3869	15053
		API	Default	-	113	229	457	898	1811
		API	4-4-32	-	110	218	442	880	1868

Acknowledgment

This work has been partially supported by the ECSEL RIA 2016 MegaM@Rt2 and AQUAS projects, and Italian project SATHERNUS.

References

[1] *GR740: The ESA Next Generation Microprocessor (NGMP)*, Accessed: 01.04.2019. `http://microelectronics.esa.int/gr740/index.html`.

[2] Kinnan, L. M., "Use of multicore processors in avionics systems and its potential impact on implementation and certification," *2009 IEEE/AIAA 28th Digital Avionics Systems Conference*, 2009, pp. 1.E.4–1–1.E.4–6. doi:10.1109/DASC.2009.5347560.

[3] Burns, A., and Davis, R. I., "A Survey of Research into Mixed Criticality Systems," *ACM Comput. Surv.*, Vol. 50, No. 6, 2017, pp. 82:1–82:37. doi:10.1145/3131347, URL `http://doi.acm.org/10.1145/3131347`.

[4] *Avionics Application Software Standard Interface: ARINC Specification 653P1-3, Required Services*, Accessed: 01.04.2019.. Aeronautical Radio, Inc. 2010-11-15. Retrieved 2013-10-20.

[5] *Avionics Application Software Standard Interface: ARINC Specification 653P2-2, Part 2, Extended Services*, Accessed: 01.04.2019.. Aeronautical Radio, Inc. 2012-06-01. Retrieved 2012-10-20.

[6] *Avionics Application Software Standard Interface: ARINC Specification 653P3, Conformity Test Specification*, Accessed: 01.04.2019.. Aeronautical Radio, Inc. 2006-10-20.

[7] *Avionics Application Software Standard Interface: ARINC Specification 653 Part 4, Subset Services*, Accessed: 01.04.2019.. Aeronautical Radio, Inc. 2012-06-01. Retrieved 2013-10-20.

[8] Li, Y., West, R. D., and Missimer, E. S., "The Quest-V Separation Kernel for Mixed Criticality Systems," *CoRR*, Vol. abs/1310.6298, 2013.

[9] *LEON/ERC32 RTEMS Cross Compilation System (RCC)*, Accessed: 01.04.2019. https://www.gaisler.com/index.php/products/operating-systems/rtems.

[10] Wasicek, A., El-Salloum, C., and Kopetz, H., "A System-on-a-Chip Platform for Mixed-Criticality Applications," *2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, 2010, pp. 210–216. doi:10.1109/ISORC.2010.43.

[11] Crespo, A., Ripoll, I., and Masmano, M., "Partitioned Embedded Architecture Based on Hypervisor: The XtratuM Approach," *2010 European Dependable Computing Conference*, 2010, pp. 67–72. doi:10.1109/EDCC.2010.18.

[12] *PikeOS Hypervisor*, Accessed: 01.04.2019. https://www.sysgo.com/products/pikeos-hypervisor/.

[13] *Wind River Hypervisor*, Accessed: 01.04.2019. https://www.windriver.com/.

[14] *INTEGRITY Multivisor Secure Virtualization*, Accessed: 01.04.2019. https://www.ghs.com/products/rtos/integrity_virtualization.html.

[15] *Virtuosity OA Certified to The Open Group FACE Technical Standard*, Accessed: 01.04.2019.. https://dornerworks.com/virtuosity-oa.

[16] *LEON3 processor*, Accessed: 01.04.2019. https://www.gaisler.com/.

[17] *GR-CPCI-GR740 Quad-Core LEON4FT Development Board*, Accessed: 01.04.2019.. https://www.gaisler.com/index.php/products/boards/gr-cpci-gr740.

[18] Ginosar, R., "Data Systems in Aerospace (DASIA)," *A survey of processors for space*, 2012, p. 5.

[19] *Benchmark Performance UT699E/700 LEON 3FT*, 2017. White Paper, Cobham.com/HiRel, November 20.

[20] Damman, C., Edison, G., Guet, F., Noulard, E., Santinelli, L., and Hugues, J., "Architectural performance analysis of FPGA synthesized LEON processors," *2016 International Symposium on Rapid System Prototyping (RSP)*, 2016, pp. 1–8. doi:10.1145/2990299.2990306.

[21] Kchaou, A., El Hadj Youssef, W., and Tourki, R., "Performance evaluation of multicore LEON3 processor," *2015 World Symposium on Computer Networks and Information Security (WSCNIS)*, 2015, pp. 1–4. doi:10.1109/WSCNIS.2015.7368277.

[22] *GR740 Technical Note on Benchmarking and Validation*, 2019-01-29. Technical Note, Doc. No GR740-VALT-0010, Issue 3.3.

[23] Tyler M. Lovelly, S. H. H., Travis W. Wise, and George, A. D., "Benchmarking Analysis of Space-Grade Central Processing Units and Field-Programmable Gate Arrays," *Journal of Aerospace Information Systems*, Vol. 15, No. 8, 2018, pp. 518–529.

[24] Lovelly, T. M., and George, A. D., "Comparative Analysis of Present and Future Space-Grade Processors with Device Metrics," *Journal of Aerospace Information Systems*, Vol. 14, No. 3, 2017, pp. 184–197.

[25] Graniszewski, A. A., W., "Performance analysis of selected hypervisors (virtual machine monitors – VMMs)," *Int. J. Electron. Telecommun.*, 2016.

[26] *Try Hyper-V Server 2012 R2. Evaluation Center. Microsoft.*, Accessed: 01.04.2019.. https://www.microsoft.com/en-us/evalcenter/evaluate-hyper-v-server-2012-r2.

[27] *Hypervisor bare-metal vSphere ESXi*, Accessed: 01.04.2019. https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.esxi.install.doc/GUID-016E39C1-E8DB-486A-A235-55CAB242C351.html.

[28] *Oracle VM Server*, Accessed: 01.04.2019. https://docs.oracle.com/cd/E50245_01/E50249/html/go01.html#gloss-server.

[29] *The Xen virtual machine monitor*, Accessed: 01.04.2019. https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/xen/.

[30] Graniszewski, A. A., W., "Performance analysis of selected hypervisors (virtual machine monitors – VMMs)," *Int. J. Electron. Telecommun.*, 2016.

[31] *Changelog for VirtualBox 6.0*, Accessed: 01.04.2019.. https://www.virtualbox.org/wiki/Changelog.

[32] *WMWare*, Accessed: 01.04.2019. https://www.vmware.com/.

[33] *Glances - An eye on your system*, Accessed: 01.04.2019. https://www.vmware.com/.

[34] Shuja, J., Gani, A., Bilal, K., Khan, A. U. R., Madani, S. A., Khan, S. U., and Zomaya, A. Y., "A Survey of Mobile Device Virtualization: Taxonomy and State of the Art," *ACM Comput. Surv.*, Vol. 49, No. 1, 2016, pp. 1:1–1:36. doi:10.1145/2897164, URL http://doi.acm.org/10.1145/2897164.

[35] S. Pinto, A. T., and Montenegro, S., "Space and time partitioning with hardware support for space applications," *2016 Data Systems in Aerospace, European Space Agency, ESA SP*, Vol. 736, 2016.

[36] A. Bufalino, S. C., G. Lisio, "New Avionics Software: an SMP Schema Managing Different Avionics Software Criticalities," *DAta Systems In Aerospace (DASIA 2017)*, Gothenburg, Sweden, 2017, p. 6.

[37] Weicker, R. P., "Dhrystone: A Synthetic Systems Programming Benchmark," *Commun. ACM*, Vol. 27, No. 10, 1984, pp. 1013–1030. doi:10.1145/358274.358283, URL http://doi.acm.org/10.1145/358274.358283.

[38] *SysGO*, Accessed: 01.04.2019. https://www.sysgo.com/.

[39] Carrascosa, E., Coronel, J., Masmano, M., Balbastre, P., and Crespo, A., "XtratuM Hypervisor Redesign for LEON4 Multicore Processor," *SIGBED Rev.*, Vol. 11, No. 2, 2014, pp. 27–31. doi:10.1145/2668138.2668142, URL http://doi.acm.org/10.1145/2668138.2668142.

[40] Federici, F., Muttillo, V., Pomante, L., Valente, G., Andreetti, D., and Pascucci, D., "Implementing mixed-critical applications on next generation multicore aerospace platforms," *EMC2 Summit at CPS Week*, 2016, p. 3.

[41] *Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments (EMC2)*, Apr. 2017. D9.6 - Space Applications Final Report.

[42] Andreetti, D., Federici, F., Muttillo, V., Pascucci, D., and Pomante, L., "Analysis and design of a Command & Data Handling platform based on the LEON4 multicore processor and PikeOS hypervisor," *DAta Systems In Aerospace*, 2017, p. 4.

[43] *Xtratum hypervisor*, Accessed: 01.04.2019. http://www.xtratum.org/.

[44] *Embedded Microprocessor Benchmark Consortium (EEMBC). CoreMark Benchmark.*, Accessed: 01.04.2019. https://www.eembc.org/coremark/download.php.

[45] *EMC2 - 'Embedded Multi-Core systems for Mixed Criticality applications in dynamic and changeable real-time environments'*, Accessed: 01.04.2019. https://www.gaisler.com/index.php/products/processors/leon4.

[46] *D9.2 - Space Applications Concept report*, 04.05.2015. Lucia Amorosi, Dario Pascucci, TASI.

[47] *D9.5 - Space Applications Detailed Description*, 17.12.2016. Elena García, Manuel Sánchez, TASE.

[48] *D9.6 - Space Applications Final Report*, 07.04.2017. Elena García, Manuel Sánchez, TASE.

[49] Pelton, J., and Jakhu, R., *Introduction to space safety regulations and standards*, 2010, pp. xli–xliv. doi:10.1016/B978-1-85617-752-8.10036-4.

[50] Modica, P., Biondi, A., Buttazzo, G., and Patel, A., "Supporting temporal and spatial isolation in a hypervisor for ARM multicore platforms," *2018 IEEE International Conference on Industrial Technology (ICIT)*, 2018, pp. 1651–1657. doi:10.1109/ICIT.2018.8352429.

[51] Crespo, A., Balbastre, P., Simó, J., Coronel, J., Gracia Pérez, D., and Bonnot, P., "Hypervisor-Based Multicore Feedback Control of Mixed-Criticality Systems," *IEEE Access*, Vol. 6, 2018, pp. 50627–50640. doi:10.1109/ACCESS.2018.2869094.

[52] Kim, H., and Rajkumar, R. R., "Predictable Shared Cache Management for Multi-Core Real-Time Virtualization," *ACM Trans. Embed. Comput. Syst.*, Vol. 17, No. 1, 2017, pp. 22:1–22:27. doi:10.1145/3092946, URL http://doi.acm.org/10.1145/3092946.

[53] Yun, H., Yao, G., Pellizzoni, R., Caccamo, M., and Sha, L., "MemGuard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms," *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2013, pp. 55–64. doi:10.1109/RTAS.2013.6531079.

[54] *SPEC's Benchmarks.*, Accessed: 01.04.2019. https://www.spec.org/benchmarks.html.

[55] *Filebench Benchmark.*, Accessed: 01.04.2019. https://github.com/filebench/filebench/wiki.

[56] *MiBench Version 1.0.*, Accessed: 01.04.2019. http://vhosts.eecs.umich.edu/mibench/.

[57] Zhong, A., Jin, H., Wu, S., Shi, X., and Gen, W., "Optimizing Xen Hypervisor by Using Lock-Aware Scheduling," *2012 Second International Conference on Cloud and Green Computing*, 2012, pp. 31–38. doi:10.1109/CGC.2012.115.

[58] *LEON4 Processor*, Accessed: 01.04.2019. https://www.gaisler.com/index.php/products/processors/leon4.

[59] *ML605 Hardware User Guide*, Accessed: 01.04.2019.. https://www.xilinx.com/.

[60] *fentISS*, Accessed: 01.04.2019. https://fentiss.com/.

[61] *Dhrystone Performance: Compiler Versions and Ground Rules.*, Accessed: 01.04.2019.. https://www.gaisler.com/doc/antn/GRLIB-TN-0015.pdf.

[62] *BCC User's Manual*, Accessed: 01.04.2019. https://www.gaisler.com/index.php/products/boards/gr-cpci-gr740.

[63] *Whetstones*, Accessed: 01.04.2019. http://www.keil.com/benchmarks/whetstone.asp.