D7.3 First Vertical, Full, System Integration and Validation Report

Contract number	688540
Project website	http://www.uniserver2020.eu
Contractual deadline	Project Month 21 (M21): 31st October 2017
Actual Delivery Date	3 November 2017
Dissemination level	Public
Report Version	1.0
Main Authors	Peter Lawthers (APM), Andreas Diavastos (UCY), Arnau Prat (SPA)
Contributors	Athanasios Chatzidimitriou (UOA)
Reviewers	Christos D. Antonopoulos(UTH), Spyros Lalis(UTH)
Keywords	Integration, StressLog, HealthLog, Predictor

Notice: The research leading to these results has received funding from the European Community's Horizon 2020 Programme for Research and Technical development under grant agreement no. 688540.

Acknowledgements

The work presented in this document has been conducted in the context of the EU Horizon 2020. UniServer is a 36-month project that started on February 1st, 2016 and is funded by the European Commission. The partners in the project are:

The Queen's University of Belfast (QUB)
The University of Cyprus (UCY)
The University of Athens (UoA)
Applied Micro Circuits Corporation Deutschland GmbH (APM)
ARM Holdings UK (ARM)
IBM Ireland Limited (IBM)
University of Thessaly (UTH)
WorldSensing (WSE)
Meritorious Audit Limited (MER)
Sparsity (SPA)

More information

Public UniServer reports and other information pertaining to the project are available through the UniServer public Web site under http://www.Uniserver2020.eu.

Change Log

Version	**Description of change**
1.0	Version for delivery to EC.

Table of Contents

Index of Figures

Index of Tables

Terminology

Term	Definition
ACPI	Advanced Configuration and Power Interface
APEI	ACPI Platform Error Interface
HEI	Hardware Exposure Interface
KVM	Kernel Virtual Machine
PMD	Processor Module
QEMU	Quick emulator
TCO	Total Cost of Ownership
SoC	System-on-Chip
VM	Virtual Machine

Table 1: Table of Terms

Executive Summary

UniServer seeks to improve the performance and energy efficiency in servers by automatically discovering the capability of the underlying hardware components to function beyond nominal operating points. By taking advantage of the extended margins inherent in processors and memories, the goal is to improve the power efficiency of ARM-based micro-servers running in the cloud or edge.

This report deliverable describes the spiral and incremental integration status as of project month M21 (October 2017) of the hardware and software components of the UniServer system. The tasks T7.1 *Vertical, full system integration* and T7.2, *Security countermeasures for micro-server architecture* contribute to this deliverable.

This is a status report only; it does not describe the interfaces in detail. For additional information, the reader is referred to the documents listed in the References section.

1. Introduction

The UniServer project targets the development of a unique methodology and infrastructure for exposing the pessimistic design margins in commercial servers and exploiting them through intelligent power, performance and reliability management schemes at the software and hardware layers. This includes the hardware, firmware, system software, virtualization, and cloud management layers.

This report describes the current status of the integration efforts of the UniServer modules. A brief description of each module will accompany the description.

The remainder of this document is structured as follows: section 2 provides an overview of the UniServer system, section 3 gives a summary of the hardware/firmware layer, section 4 describes the important UniServer modules in the system, section 5 discusses the application layer, and section 6 presents the conclusions and directions for future work.

2. System Overview

The UniServer system can be logically divided into three parts:

Level	Description
Hardware/Firmware	X-Gene2 based “Tigershark” systems or X-Gene3 based “Osprey” systems and firmware
System Software	Linux OS, UniServer system modules, virtualization extensions such as KVM/Qemu, and OpenStack cloud infrastructure
Application	Applications running on bare metal hardware or in the context of a VM

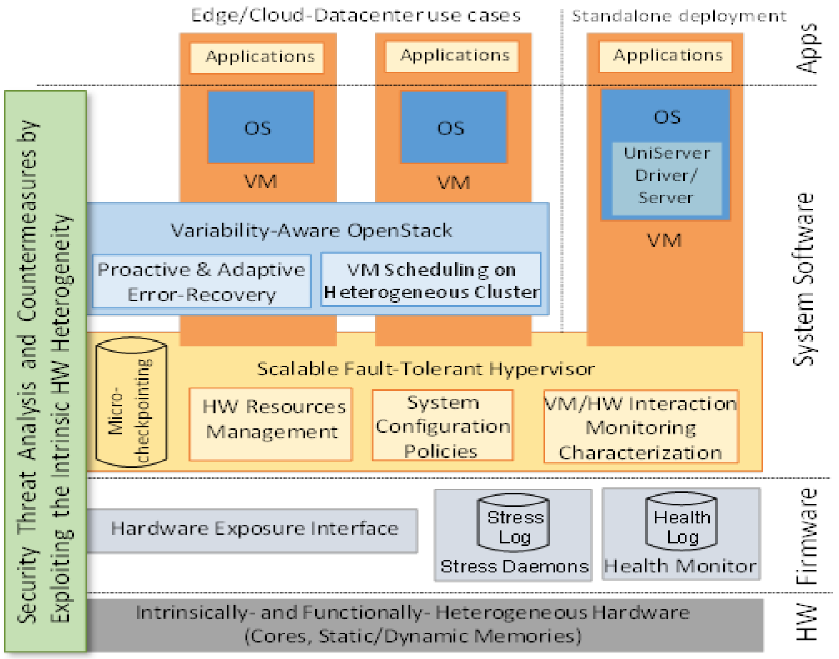
Table 2: Uniserver levels

The newly developed UniServer components are the *Health Monitor* and the *Stress Monitor,* modules. These components do not exist in standard system software distributions, and become part of the System Software level in the above table. In addition, there has been extensive work focused on the virtualization layer.

The hardware components are stress tested during a pre-deployment phase using various stress methods, consisting of a combination of viruses and real application benchmarks. The *Health Monitor* [4] continuously monitors the health status of the hardware under a particular voltage/frequency/refresh rate (V-F-R) point and provides the information to other layers in the system via vectors describing the performance, power, temperature, and any incurred errors.

The *Stress Monitor* [5] is responsible for testing the hardware components, producing a log file (the StressLog) containing a summary of the execution, including observed errors (read from the Health Monitor log), observed SDCs, application crashes, etc. Typically, the Stress Monitor is called by the Predictor using different V/F configuration parameters, different viruses/benchmarks, etc.

The Predictor consumes the logs produced by the Stress Monitor and builds probability failure models that will then be used to predict the hardware behaviour under different operating points. Given these models, the upper layers (i.e. the Hypervisor), can ask the Predictor for optimal operating points based on the expected reliability of the system.


Edge/Cloud-Datacenter use cases
Standalone deployment
Applications
Applications
Applications
Apps
OS
VM
OS
VM
OS
UniServer Driver/ Server
VM
Security Threat Analysis and Countermeasures by Exploiting the Intrinsic HW Heterogeneity
Variability-Aware OpenStack
Proactive & Adaptive Error-Recovery
VM Scheduling on Heterogeneous Cluster
System Software
Micro-checkpointing
Scalable Fault-Tolerant Hypervisor
HW Resources Management
System Configuration Policies
VM/HW Interaction Monitoring Characterization
Hardware Exposure Interface
Stress Log
Stress Daemons
Health Log
Health Monitor
Firmware
Intrinsically- and Functionally- Heterogeneous Hardware (Cores, Static/Dynamic Memories)
HW
Figure 1: Detailed UniServer Layering

The UniServer layering is shown in the above diagram.

The Hypervisor attempts to limit the effects of potential faults in higher software layers by reconfiguring the system to operate within safe margins and isolating problematic processing and memory resources that affect the VMs. This is achieved by utilizing the information delivered by the Health Monitor and the Predictor models and developing a new set of configuration properties. Finally, when aging effects are being observed by the Hypervisor via the Health Monitor, the Hypervisor can ask the Predictor to rebuild the models which in turn, will make the Predictor to call the Stress Monitor and restart the training cycle.

The virtual machines are all controlled by the cloud management software, OpenStack. OpenStack controls multiple nodes and tries to minimize the energy footprint of the system, while at the same time respecting Service Level Agreements. It queries the Hypervisor to identify the tradeoffs between energy, performance and reliability, and sets nodes at appropriate configurations and schedules VMs.

3. Hardware/Firmware Layer

The interface between the X-Gene processor and the system software (Linux) is described in detail in D4.1 [6]. The API includes a notification interface, the intent of which is to provide an indication of failures during undervolting. However, undervolting testing [7] has shown that the first indication of an error is often silent data corruption, requiring additional checks in the upper layers.

3.1 Status

The X-Gene2 “Tigershark” platforms have been deployed since project month M5; the X-Gene3 “Osprey” platforms have been delivered as of project month M20.

As of project month M21, the API is fully functional on the X-Gene2 based “Tigershark” platform. This is an I2C based interface running on Centos based Linux 4.3.0. The API is also in testing on X-Gene3 based “Osprey” platforms. The notification API is layered on top of the standard Linux APEI error reporting interface, leveraging the existing firmware and reducing the necessity of custom firmware builds.

The primary consumer of the HEI API is the Health Monitor, and the integration between the Health Monitor and hardware/firmware has been completed.

4. System Software Layer

4.1 Linux Platform

There are two primary software platforms for UniServer. On the X-Gene2 based "Tigershark" platforms, a Centos 7.2 based distribution has been used up until project month 21 (M21). This uses a Linux 4.3.0 kernel. For the remainder of the project, the Tigershark platform can use either the Centos7.2 distribution, or a Centos7.4 based platform, featuring a Linux 4.11 kernel. The X-Gene3 based "Osprey" platform only supports the Centos7.4 distribution.

4.1.1 Status

Along with the hardware platforms, a Linux distribution supporting the hardware and providing the HEI API has been delivered for both "Tigershark" and "Osprey" systems.

4.2 System Modules

An overview of the modules in the system software layers is shown in the following diagram.

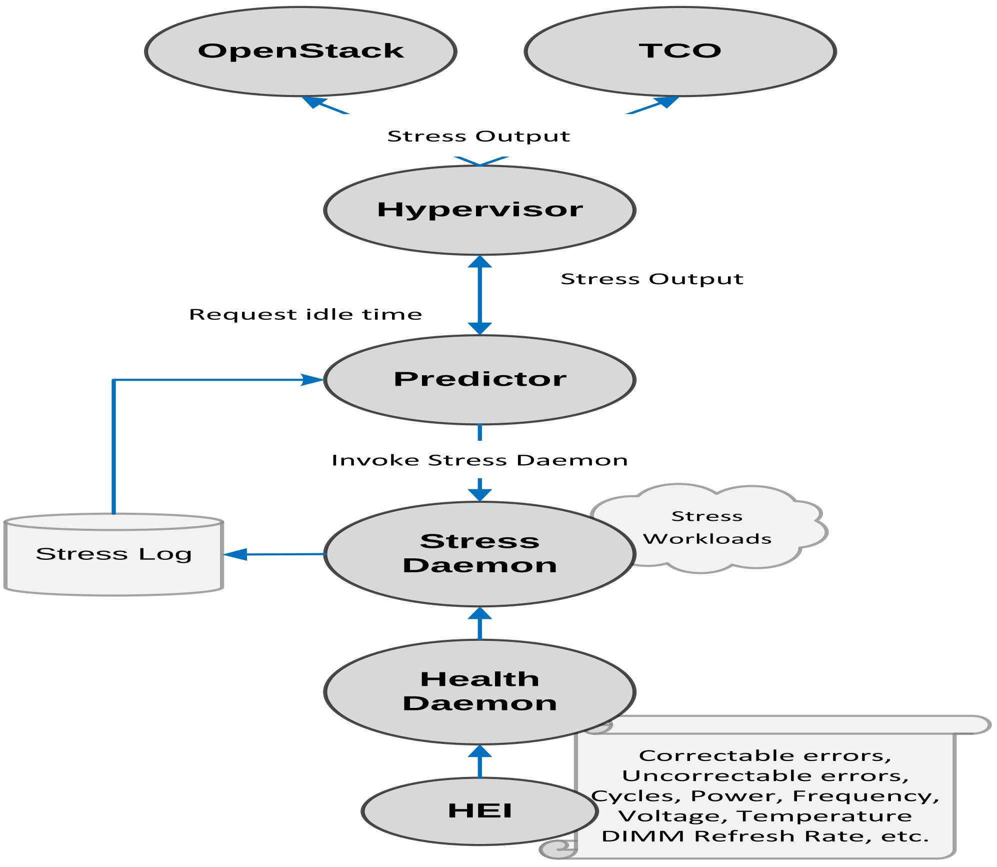


Figure 2: Module Layering

4.3 HEI

The Hardware Exposure Interface ([6]) provides two separate functions:

- Exposure of architecture-specific sensors and registers via an I2C register map

- Notification of processor error events

The I2C register map allows upper layer modules to monitor and control power and frequency for the processor and DIMMs. The event notification API provides a straightforward interface for the receipt of error events. The HEI has been implemented for both X-Gene2 and X-Gene3. The implementation is quite different, as the firmware for each chip shares almost nothing in common.

4.3.1 Status

The HEI has been implemented and integrated with the Health Monitor for X-Gene2; an initial version supporting the X-Gene3 platforms has been delivered, and testing is underway on a version supporting all the I2C registers specified in the D4.1 document.

4.4 Health Monitor

The Health Monitor ([4]) receives error events from the hardware/firmware via the HEI notification interface. The Health Monitor monitors the state of the system producing a log that is consumed by the Stress Monitor and the Predictor. The Health Monitor also features an error testing interface, to ensure that the errors that can be reported are correctly parsed by the other modules in the system. In addition, the Health Monitor keeps track of some statistical information regarding health-related incidents and provides these statistical data through an internal text-based API, on an "on-demand" scheme.

4.4.1 Status

As of project month 21 (M21), the Health Monitor has been integrated into the system, and the interfaces to the Predictor and Stress Monitor are being finalized, along with the "on-demand" serviced, as these are described on the D4.2 . An additional interface for reporting high-level information on the log (by the Hypervisor) is also being integrated, without however affecting the overall development plan, which is according to schedule.

4.5 Stress Monitor

The Stress Monitor ([5]) can be triggered at any time to run stress tests to produce logs containing the details about the observed system’s behavior (such as aging effects). The tests can be configured in a way that combinations of different benchmarks/viruses, bound to different cpu cores, and with different V/F parameters can be executed. The module is typically called by the Predictor, which uses the logs to build the models to predict the behaviour of the system at different operating points. Alternatively, it can be triggered by the system’s administrator or the Hypervisor if aging effects are being detected via the Health Monitor.

4.5.1 Status

The Stress Monitor has been integrated into the system, and the interfaces to the Predictor are being finalized. Additionally, it has been integrated with the Health Monitor so it can read the errors that occur in the system during the execution of the stress tests.

4.6 Predictor

The Predictor is a low-level handler that gathers different information from the hardware using the Health Monitor and Stress Monitor statistics and uses low complexity (thus fast) but effective predictive and learning techniques to predict failures. The goal is to be able to allow the system to execute in states below nominal margins while monitor and take action at low level as to avoid system failures. Regarding the prediction techniques to be used, a machine learning algorithm may be initially trained using the data observed during the system characterization and subsequently predict how to adjust V-F states and DRAM refresh rate to maximize energy efficiency without hurting performance and without compromising availability. Machine learning techniques seem to be a good match for the proposed goal since they are, by design, capable of detecting when they need to be retrained, effectively when they are performing poor due to non-representative training or when system behavior changes either due to load changes or environmental conditions. The constraint of this technique is not to add any significant overhead to the execution or power consumption of the system.

4.6.1 Status

Draft designs of the Predictor module have been implemented and discussed with all partners. This discussion is still on-going as to define its functionality and its interaction with all other software components of the UniServer platform. To this end, the interfaces between the Predictor and other software components (Hypervisor, Stress Monitor) have already been defined and implemented. The first implementation of the Predictor module is finished with the Predictor gathering information from the logs and creating a database with failure probabilities for different operating points. These values can be returned to Hypervisor when requested.

4.7 Hypervisor

The hypervisor and virtual machine support facilities are comprised of the KVM modules, Qemu user-space support, and `libvirt`. These modules are standard Linux components that have been enhanced for UniServer.

The hypervisor relies on the Health Monitor and other components to track the current reliability and stability of the system. For example, if a core or set of cores is deemed unreliable, then key processes and/or OS & Hypervisor functionality can be migrated to reliable cores. Memory can also be partitioned into reliable and unreliable domains. The hypervisor itself will be restricted to allocating memory only from the reliable domain.

4.7.1 Status

The hypervisor, Qemu, and `libvirt` have been augmented with the UniServer enhancements; the integration with OpenStack is being finalized with the implementation of the interfaces described in D5.3. The integration and bidirectional information flow with Health Monitor and the Predictor is finished.

4.9 OpenStack

OpenStack provides the framework for managing multiple virtual machines in the UniServer environment. The primary components of OpenStack are as below:

1. **Nova:** responsible for provisioning and management of virtual machines on compute nodes
2. **Glance:** stores the images that are used as the disk templates for booting VMs
3. **Keystone:** manages authentication and authorisation for invoking OpenStack commands
4. **Swift:** is an object storage system wherein uniquely identified data items can be manipulated independent of their location
5. **Cinder:** allocates filesystem blocks that can be attached to VMs as extended disk storage
6. **Neutron:** provides networking services for OpenStack components as well as VMs
7. **Ceilometer:** provides telemetry services to keep track of resource usage and performance
8. **Horizon:** is a web-based user interface to interact with an OpenStack installation

The relationship between the hypervisor and OpenStack is shown in the following diagram:

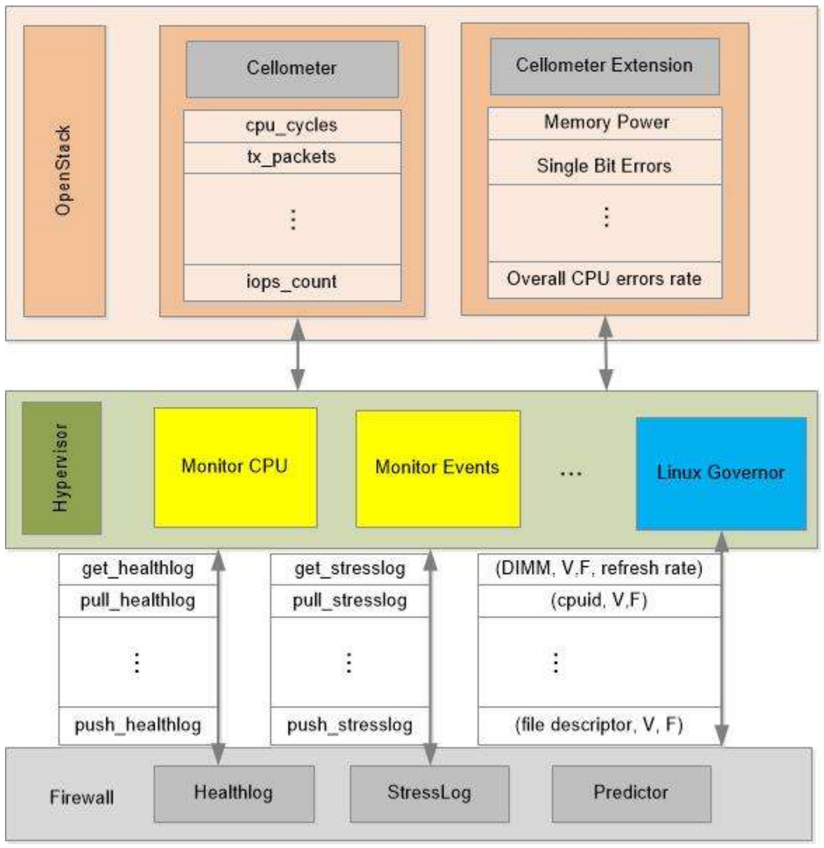


Figure 3: Hypervisor and OpenStack relationship

The enhancements to OpenStack are critical for obtaining the information required by the cloud middleware to improve the performance, reliability, and the energy consumption of the data center. To this end, the two components in OpenStack that are of particular interest for UniServer are the telemetry (Ceilometer) and compute (Nova). Ceilometer provides detailed performance and utilization measurements that are related to the physical and virtual resource, while Nova provides the computing fabric controller for the cloud and manages the virtual machines running in the data center.

4.9.1 Status

The Ceilometer component has been augmented with UniServer specific interfaces as described in [1] and [3]. This work has focused on enhancing libvirt to allow UniServer specific parameters to be taken into consideration during OpenStack operation.

An X-Gene2 based “Merlin” system running a generic Centos7 release has been the initial staging and development platform. This has been used to develop the libvirt extensions between OpenStack and the other components of the UniServer system so that OpenStack may take advantage of the extended operating parameters of UniServer.

5. Application Layer

There are three target applications for UniServer, listed in the table below.

Application	Provider
Wireless jamming detection	WSE
Trade confirmation	Meritorious
Social CRM and Social TV	SPA

Table 3: UniServer target applications

The applications do not have any specific interfaces to the UniServer system; they remain agnostic about the type of platform they are executing on.

5.1 WSE Wireless Jammer Detector

The SDR Jammer Detector is a wireless security component of the Denial of Service (DoS) Sensing solution. The detector identifies jamming signal threats that aim to generate DoS attacks by interfering with wireless network communications. For this purpose, this solution implements a smart sensor that detects threats and communicates detection events to visualization software so that users can easily identify the type of jamming signal generating the attack.

5.1.1 Status

The WSE Wireless Jammer Detector application has been the first priority. It is the showcase of the UniServer demo, scheduled for project month 22 (November 2017).

5.2 Meritorious Trade Confirmation

European Markets Infrastructure Regulation (EMIR) came into force on 16 August 2012, and introduced requirements aimed at improving the transparency of Over-The-Counter (OTC) derivatives markets and to reduce the risks associated with those markets. In order to achieve this, EMIR requires that OTC derivatives meeting certain requirements be subject to the clearing obligation and for all OTC derivatives that are not centrally cleared that risk mitigation techniques apply. In addition, all derivatives transactions need to be reported to Trading Repositories (TRs).

The reporting of a derivative transaction involves any daily modifications/updates of the transaction until the termination of the derivative contract. This requires the processing and validation of a large amount of information and handling sensitive client's data.

5.2.1 Status

Work has begun in porting the Polaris Platform and the Micro Polaris Benchmark to the ARM based UniServer platform and is approximately 25% complete.

5.3 SPA: SocialCRM/SocialTV

The two SPA apps (Social CRM and Social TV) share the same three architectural components:

The **social network server** component which runs on a large data center

The **social analytics components** that run on client's premises and which process data from the server regarding specific subscribed events

The **web app**, which also runs on client's premises and that connects to a database with the results of the social analytics component to allow their exploration

5.3.1 The social network server component

The social network server uses synthetically generated data, which simulates a real social network in time. This dataset is bulk loaded into the server to set it into a working state, and spans several years of social network activity. Additionally, the synthetic datasets also contain "update" streams, which basically consist of the update activity in the social network (new messages, new friends, etc.) in the subsequent months in the simulation that have not been bulk loaded, but are fed into the driver which is responsible for issuing these updates at runtime. Also, parameters are provided to the driver to perform read queries.

5.3.2 The social analytics component

The analytics component, which is installed in client's premises, receives periodic batches of updates on those specific events tracked. For instance, imagine a client A wants to track the activity related to Donald Trump. The social network server will buffer this activity and from time to time or when a buffer is filled it will send that to the client (the social analytics component). From time to time, the client will process the data and combine (or not) with the previous data, and update its snapshot of information with the most recent data received from the server. The type of processing is like finding influencers, detecting communities, etc.

5.3.3 The web app component

For the web app, we must guarantee that this is interactive and available. This is similar to the social network server. We could measure the latency of the requests and the user experience. Assuming a workday of 8 hours, an SLA of 99% would guarantee a non-availability of at most 5 minutes per day. This seems reasonable.

5.3.4 Status

The Social Analytics component for the SocialCRM application has been developed and integrated with the UniServer platform. RPM packages to be delivered via the CentOS package manager has been produced for an easy installation. The Web app for the SocialCRM component has also been developed, although final integration is on the works. The development of the Social Network Server is in progress.

.

6. Conclusions and Future Work

This report deliverable has described the UniServer system at the hardware, system, and application layers in the initial period. The modules at each layer have been discussed, as well as the level of integration. The remainder of the project will focus on integrating all components on the target hardware, both X-Gene2 and X-Gene3 based platforms. OpenStack will focus on enhancing the Nova Component with UniServer specific parameters. The deployment of OpenStack on the X-Gene3 based “Osprey” systems will allow for the entire stack, from hardware/firmware to application, to be exercised.

.

7. References

[1] D5.1 1st Report on Hypervisor /System / Software Interface
[2] D5.3 2nd Report on Hypervisor / System / Software Interface
[3] D6.1 OpenStack Support for UniServer
[4] D4.2 HealthLog Specifications and Interface
[5] D4.3 StressLog Specification and Interface
[6] D4.1 Hardware Exposure Interface (HEI) and Error Handlers Specification
[7] G. Papadimitriou et al., “Harnessing voltage margins for energy efficiency in multicore CPUs”. MICRO 2017

[END OF DOCUMENT]