An Energy-Efficient and Error-Resilient Server Ecosystem Exceeding Conservative Scaling Limits

An Energy-Efficient and Error-Resilient Server Ecosystem Exceeding Conservative Scaling Limits

Georgios Karakonstantis
Konstantinos Tovletoglou
Lev Mukhanov
Hans Vandierendonck
Dimitrios S. Nikolopoulos
Queen's University Belfast
Peter Lawthers
A.M.C.C. Deutschland

Panos Koutsovasilis
Manolis Maroudas
Christos D. Antonopoulos
Christos Kalogirou
Nikos Bellas
Spyros Lalis
University of Thessaly

Srikumar Venugopal
IBM Research - Ireland
Arnau Prat-Perez
Sparsity
Alejandro Lampropulos
Worldsensing
Marios Kleanthous
Meritorious

Andreas Diavastos
Zacharias Hadjilambrou
Panagiota Nikolaou
Yiannakis Sazeides
Pedro Trancoso
University of Cyprus

George Papadimitriou
Manolis Kaliorakis
Athanasios Chatzidimitriou
Dimitris Gizopoulos
University of Athens
Shidhartha Das
ARM Ltd.

ABSTRACT

The explosive growth of Internet-connected devices will soon result in a flood of generated data, which will increase the demand for network bandwidth as well as compute power to process the generated data. Consequently, there is a need for more energy efficient servers to empower traditional centralized Cloud data-centers as well as emerging decentralized data-centers at the Edges of the Cloud. In this paper, we present our approach, which aims at developing a new class of micro-servers – the UniServer - that exceed the conservative energy and performance scaling boundaries by introducing novel mechanisms at all layers of the design stack. The main idea lies on the realization of the intrinsic hardware heterogeneity and the development of mechanisms that will automatically expose the unique varying capabilities of each hardware component within commercial micro-servers and allow their operation at new extended operating points. Low overhead schemes are employed to monitor and predict the hardware behavior and report it to the system software. The system software including a virtualization and resource management layer is responsible for optimizing the system operation in terms of energy or performance, while guaranteeing non-disruptive operation under the extended operating points. Our characterization results on a 64-bit ARMv8 micro-server in 28nm process reveal large voltage margins in terms of Vmin variation among the 8 cores of the CPU chip, among 3 different sigma chips, and among different benchmarks with the potential to obtain up-to 38.8% energy savings. Similarly, DRAM characterizations show that refresh rate and voltage can be relaxed by 43x and 5%, respectively, leading to 23.2% power savings on average.

1. INTRODUCTION

The number of intelligent Internet-connected devices is growing daily and will soon be in the orders of tens of billions, forming the Internet of Things (IoT). Each of these devices is pushing data to the Internet and this data is expected to reach 24.3 exabytes in 2019 [1]. This rapid data growth will put a lot of pressure on the current Internet infrastructure and centralized data-centers, which are already oversubscribed. Coping with this imminent data flood requires not only enhancement of the processing capabilities of the current servers but also rethinking of the way we communicate and process data across the Internet.

A recently introduced approach that has the potential to ensure the viability and scaling of the Internet in the IoT era is *Edge* computing, which evangelizes the execution of services closer to the data sources [2] helping to reduce application latency between the end user and the data-center. Realizing such a paradigm requires the design of new server ecosystems that can be deployed closer to the data sources without the need of any expensive cooling or power infrastructure. This is contingent on designing such ecosystems with substantially improved energy efficiency than the current state-of-the-art servers without compromising performance, availability, programmability, reliability and security properties of the existing cloud data-centers.

However, realizing such server ecosystems is extremely challenging due to the stagnant voltage scaling (the most effective power saving knob), and the worsening process variations [3], [4] that nanometer circuits are experiencing. In fact, as transistors are being pushed to the atomic scale, it is becoming very difficult to fabricate circuits with the expected power and performance specifications leading to large static and dynamic variations [3-9].

To cope with the significant hardware variability and avoid the risk of system failures, manufacturers try to hide it from the system software by adopting pessimistic voltage and frequency margins/guard-bands based on the worst-case manufactured chips and assumed scenarios [3, 5-9]. However, such guard-bands end-up forcing the circuits to work less efficiently than they could, essentially constraining the power and performance of all the manufactured circuits based on the worst-case parts. Such margins are becoming more prominent with the use of more cores per chip, the increased voltage droops [5], reliability issues at low voltages (Vmin) [6], and core-to-core variations. Indicatively, recent measurements in ARM processors indicated that more than 30% timing and voltage margins were adopted in 28nm chips [4].

Realizing that the power and performance overheads imposed by the current pessimistic design paradigm is unavoidable, in this paper we introduce a radical approach that plans to turn the table around by treating the intrinsic hardware heterogeneity as an opportunity and not as a problem. In particular, in UniServer we put forward the following question: Why allow the worse margins of fabricated chips to artificially constrain the performance and energy of today's systems? The reality is that each manufactured processor and each memory module is inherently different and lies on a distinct performance bin as depicted in Figure 1. Based on such observation, the UniServer approach plans to substitute the existing conservative margins with the real capabilities of each individual core and memory-array. This will enable us to exceed the energy and performance scaling boundaries

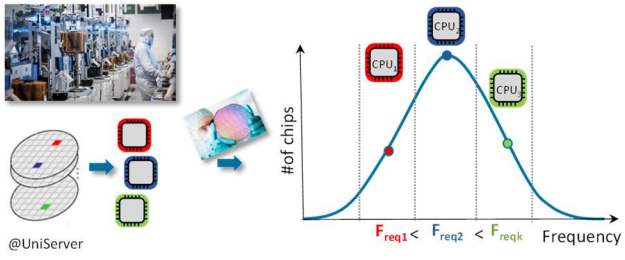

Figure 1: Each manufactured chip is intrinsically different in terms of capabilities

The presented research effort has received funding from the European Community's Horizon 2020 programme under grant no. 688540 (UniServer). - http://www.uniserver2020.eu/

adopted in commercial servers. The UniServer project introduces the following technical innovations at all system layers:

i) automatically reveal the possible Extended Operating Points (EOP) (*i.e.,* voltage, frequency, refresh rate) of each hardware component (*i.e.* cores and memories);
ii) monitor and predict the operating status of the underlying hardware components by introducing low-level daemons;
iii) optimize the system operation by adjusting the power, performance, reliability trade-off based on the enhanced policies;
iv) enable monitoring of the hardware status by all layers of the system software by extending existing interfaces;
v) enhance the fault tolerance of all layers of the system software by providing sufficient protection to critical software structures;
vi) adapt software packages for virtualization and resource management to leverage EOP on next-generation servers;
vii) develop a tool for estimating the Total Cost of Ownership (TCO) gains against other solutions that can be achieved by deploying UniServer in Edge and cloud data-centers and
viii) analyze security threats in servers operating under the new EOP and provide low cost countermeasures.

The purpose of this paper, is to introduce the proposed approach and present our results within the first year of the project.

The rest of the paper is organized as follows. Section 2 presents the cross layer UniServer approach. Section 3 discusses the innovation at the hardware and firmware layer. Section 4 discusses the approach followed at the System Software. Section 5 presents the targeted improvements and presents the obtained results, while Section 6 discusses with the state-of-the-art. Finally, conclusions are drawn in Section 7.

2. THE UNISERVER APPROACH

Figure 2 depicts the different layers of the UniServer ecosystem. The most fundamental idea of the project lies on the hypothesis that each hardware component (*i.e.* core, cache, DRAM) may have intrinsically different capabilities in terms of energy, performance and reliability.

HW/Firmware. Starting from the low layers, we develop techniques that aim at revealing new EOP for each hardware component based on the component's true capabilities. This is achieved by stress-testing the hardware components during a pre-deployment phase under different points using various stress kernels. During deployment, a *HealthLog* daemon is monitoring online the health status of the hardware under any used voltage/frequency/refresh rate (V-F-R) point and informs the system software by propagating information vectors about the performance, power, temperature, and any incurred errors. Moreover, another Linux daemon, the *StressLog*, is responsible for periodic offline, on-demand stress testing of the hardware components and for producing an output vector containing the new safe system V-F-R margins that will be suggested to the software (i.e. Hypervisor) for future usage. It also produces log files recording errors (correctable or uncorrectable), system configuration values, sensor readings and performance counters. Using the information provided by the *HealthLog* and *StressLog,* the *Predictor* develops probability failure models and tries to predict the hardware behavior under any operating point and eventually helping the system software to decide on the optimum configuration.

System Software. UniServer targets a wide range of use cases, ranging from deployments in remote locations close to the end users to deployments in cloud data-centers. To facilitate such diverse use cases, the UniServer platform must be equipped with a complete software stack that can efficiently manage any compute and storage resources by offering easy installation, migration and replication of tasks, either at the node or server-rack level. To this end, state-of-the-art software packages for virtualization (Hypervisor) and resource management (OpenStack) are being adopted. Such packages, apart from managing the Virtual Machines (VMs) at the node level (Hypervisor) and the resources at a rack/data-center level (OpenStack), they are also being enhanced for optimizing the system operation and the available

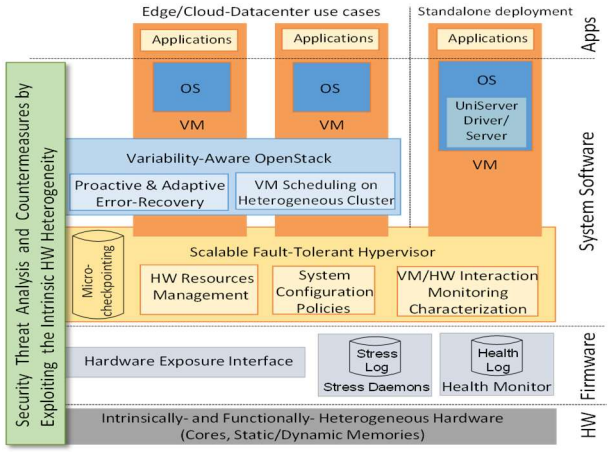

Figure 2: UniServer - Cross-Layer Error-Resilient Ecosystem

resources by fine tuning the extended V-F-R points. In particular, the Hypervisor will aim at limiting the effects of the potential faults to 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 *HealthLog/StressLog/Predictor* daemons and developing a new set of configuration properties. The optimization of operations at the EOP in UniServer is guided by the system requirements of the end-user for each VM, which are typically communicated to the Cloud provider through Service Level Agreements (SLAs). These workload-specific requirements reflect the key metrics of interest based on which OpenStack manages the nodes that constitute any data-center. Note that in UniServer an additional node reliability metric is added to the traditional metrics of interest, which are node availability, utilization and energy usage. Altogether, these metrics will help in system optimization. The system optimization will be also assisted by developing a tool for estimating the potential TCO gains that can be achieved by various configuration properties of the platform and deployments on Cloud or Edge environments.

UniServer Chassis. The developed technologies are being ported on a state-of-the-art 64-bit ARM based Server-on-Chip (SoC), the X-Gene 2. The micro-server consists of eight 64-bit ARMv8-compliant cores, grouped in 4 Processor Modules, which have a separate 32KB instruction and 32KB data caches for each processor (Parity protected) and unified 256KB L2 cache (ECC protected). The 8MB L3 cache is shared across the whole chip and is ECC protected. There are 4 available memory channels supporting ECC protected DDR3-1866 with up-to 512GB capacity. This board allows us to control independently the voltage and the frequency of the cores, caches, DRAMs/.

Note that the developed technologies will not be tied only on the particular platform and special consideration is given to enable their seamless integration with other servers.

3. EXPOSING MARGINS AND MONITORING HW BEHAVIOR

UniServer uses the following technical approach for revealing optimistic margins. Firstly, at the pre-deployment stage, the system goes through a batch of stress-tests to determine the more efficient but safe per-component margins. Secondly, at normal operation in the field, a daemon is constantly recording any possible errors (even if correctable) to fine-tune the margins after deployment. If the number of errors rises above a certain threshold a new stress-test cycle may be triggered to determine new efficient safe margins. This is useful to better adapt to the workloads and also to the aging of the system. Thirdly, during runtime a predictor daemon is running to observe different metrics and advise the Hypervisor on possible execution modes (e.g. high-performance or low-power).

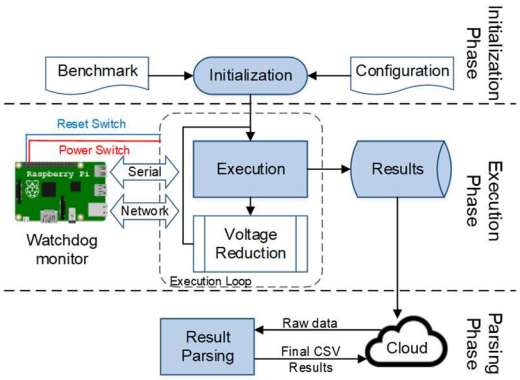

Figure 3: Characterization framework layout.

A. Revealing the margins within on-board components

Heterogeneity exists among cores located on the same chip, DRAM and cache memory banks. Each resource may perform better or worse than others but certainly not as any other similar resource on the board. In UniServer, we characterize each core and memory bank individually. For example, for each cache memory bank UniServer reveals the minimum voltage that allows correct operation. This information is exposed to software and is exploited towards better energy-efficiency.

To this end, we developed an automated characterization framework, as shown in Figure 3, (1) to identify the target system's limits when it operates at scaled voltage and frequency conditions, and (2) to log the effects of a program's execution under these condition. As shown in Figure 3, the characterization framework consists of three phases: initialization, execution, and parsing. During the initialization phase, a user can declare a benchmark list with corresponding input datasets to run in any desirable characterization setup. The characterization setup includes the voltage and frequency (V-F) values on which the experiment will take place and the cores where the benchmark will be run. The execution phase consists of multiple runs of the same benchmark, each one representing the execution of the benchmark in a pre-defined characterization setup. The set of all the characterization runs of the same benchmark with different characterization setups represents a campaign. In the parsing phase of our framework, all log files that are stored during the execution phase are parsed in order to provide a fine-grained classification of the effects observed for each characterization run. A similar flow is also followed for the characterization of on board DRAMs under various supply-voltages and refresh rates.

B. Stress-test development

To characterize the hardware components, we stress the underlying cores and memories using diagnostic viruses which either are based on known stress kernels, or are being generated by genetic algorithms [10], or are based on application workloads. Such viruses represent pathogenic worst-case scenarios that is unlikely to be encountered in real-life workloads targeting to cause maximum voltage noise, power consumption and error rates.

C. HealthLog Daemon

Operating outside the nominal values may introduce hardware errors during the system's lifetime. Thus, there is a need for a runtime mechanism that will monitor the system and report errors occurring during uptime. Such mechanisms already exist for different platforms but important information is missing. Therefore, in UniServer we are extending existing knowledge to create a UniServer-specific monitoring mechanism. We have extended the error reporting capabilities of existing mechanisms with system configuration values, sensor readings and performance counters. We call this mechanism the *HealthLog* monitor that records runtime system metrics in the form of an information vector, stored in a system logfile. The *HealthLog* monitor interact and exchange information with higher system layers (e.g. the *Predictor* and the *Hypervisor*). The *HealthLog* monitor provides two types of services: (a) Event-driven services, where it will collect information based on event occurrences in the system (e.g. errors) and (b) On-demand services, where the monitor will respond to requests from higher layers for specific information. Every entry consists of 4 columns and each column is separated by a delimiter. The first column is an incremental identifier for the log entry, followed by the date, entry type and finally entry details.

D. StressLog Daemon

It is expected that the EOP may need to be updated several times over the lifetime of a server due to the aging effects of the machine or unexpected errors observed. Therefore, a mechanism is needed, to produce new nominal values that will still guarantee the safe operations of the server. This mechanism stress-tests the machine using predefined applications and compute new safe operating V-F-R margins. We call this mechanism the *StressLo*g monitor.

The *StressLog* monitor is spawned either periodically during a machine's lifetime (e.g. every 2-3 months) or is triggered by higher system layers in the case of anomalous machine behavior. In this case, the machine being tested will be taken offline and as soon as the monitor receives the input stress target parameters from the higher system layers, it will initiate the stress test scenarios. The *StressLog* monitor also includes a workload suite, consisting of different benchmarks and test kernels that either represent real-life applications or are hand-coded to stress specific components of the system. During a stress test, the *HealthLog* monitor executes in parallel to record system events (errors, system values, sensors and performance counters). The *StressLog* monitor takes the output of the *HealthLog* and wraps the needed information into a vector to be passed to the higher layers.

E. Predictor

The predictor is a software that will utilize offline characterization data along with predictive techniques e.g. machine learning, linear regression to predict the probability of failure for non-nominal voltage frequency states and DRAM refresh rates. Particularly, given availability constraints and desired number of cores and operating frequencies; the predictor estimates the most energy efficient voltages and DRAM refresh states that don't violate the given constraints.

One of the main challenges that predictor has to address, is the ability to capture the effect of combining probabilities of failure across multiple components, namely CPU, SoC and DRAM. Also, predictor will have to be able to re-adjust the probabilities of failure on the fly based on the online system operation. For instance, predictor should be able to adapt according to StressLog daemon run results as well on unexpected emergencies that might happen during system operation like crashes.

4. MANAGING OPERATION AT EXTENDED MARGINS

A. Virtualization

UniServer follows a Hypervisor-based approach based on KVM, to leverage all benefits of virtualization, such as easier deployment, administration, replication and migration, which are necessary for the targeted data-centers at the Edge of the Cloud.

In the context of UniServer, the Hypervisor has additional roles. It is responsible for creating an appropriate execution environment for VMs by manipulating the power / performance / reliability tradeoffs in an educated and safe manner. Specifically, it sets the system at a just-right configuration, which reduces the power footprint of each node by eliminating unnecessary hardware guard-bands, without introducing negative effects on the services running within the VMs. As discussed earlier, the best configuration depends on a number of different parameters, including the characteristics of application software, the capabilities of the specific hardware parts at the specific time and under the specific environmental conditions, as well as the quality of service (QoS) requirements introduced by the cloud management framework (OpenStack).

Despite applying sophisticated configuration policies within the limits specified by the *StressLog*, *Predictor,* sporadic errors may still inadvertently occur due to the elimination of guard-bands. The Hypervisor needs to protect the whole system from catastrophic failures. Beyond selecting a realistic hardware configuration, the

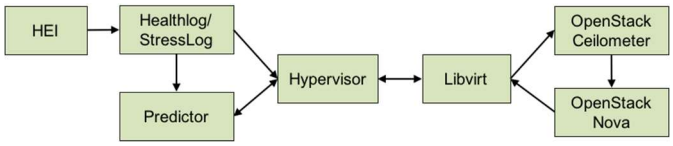

Figure 4: Flow of information among the UniServer components

Hypervisor isolates problematic processing and memory resources experiencing high error rates, as reported by the *HealthLog*. Being the lowest level of system software, the Hypervisor itself needs to be resilient to errors. We use fault injection to characterize the sensitivity of Hypervisor data structures both at the kernel- (KVM) and user-level (QEMU), in order to enable educated, selective protection. Moreover, we break the SMP assumptions of Linux/KVM and implement migration of critical system operations to reliable cores, in order to increase its resilience on top of mixed reliability hardware.

B. Resource Management - OpenStack

The next layer of software is the OpenStack [11] which is widely used open source middleware for cloud setups, and it pairs well with the popular enterprise and open source technologies. Our extended version of OpenStack, includes support for monitoring VMs and determining their dynamically changing characteristics and virtual resource utilization at a finer granularity than the existing state-of-the-art. In particular, the Ceilometer component of OpenStack gathers various data about the health and performance of the underlying physical and virtual resources in the data center as shown in Figure 4 with the help of Hypervisor who gathers the requested information through the StressLog and HealthLog daemons.

OpenStack Nova has the responsibility to manage the resources of the physical hosts, to map and deploy incoming VMs to available nodes, and to maintain the 'good' health of the running VMs. In the context of UniServer, Nova is extended to switch nodes into more power-efficient voltage-frequency configurations. This could involve running a node at the extended margins which could lead to increased probability of faults affecting the applications running inside the VMs. Therefore, the VM scheduler within Nova is being extended to consider the sensitivity of applications to system errors before mapping VMs to nodes running in different configurations.

5. Results and Saving Projections

In this section, we present our pre-deployment characterization results obtained in the initial phase of the project for the on board cores and DRAMs and different Hypervisor structures within our ARMv8 server prototype for a variety of benchmarks. Furthermore, we analyze the estimated TCO for indicating the potential improvements.

A. Characterization of CPUs

We experimentally obtain the Vmin values of the 10 SPEC CPU2006 [12] benchmark on the three X-Gene 2 chips (TTT, TFF, TSS) [13],

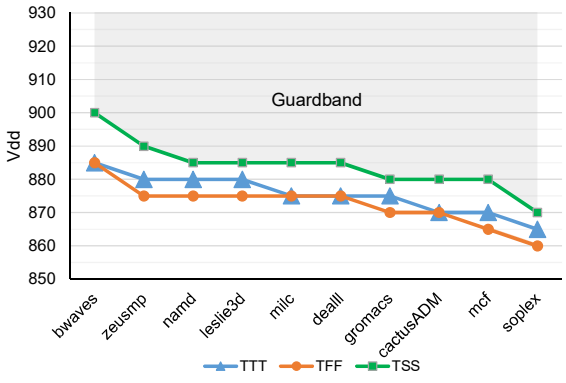

Figure 5: Vmin results at 2.4 GHz for 10 SPEC CPU2006 programs on 3 different X-Gene 2 chips (TTT, TFF, TSS).

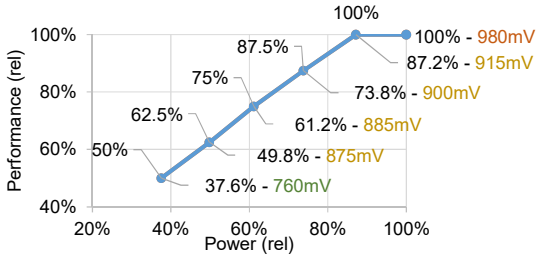

Figure 6: Tradeoffs for a workload of 8 benchmarks.

running the entire time-consuming undervolting experiment ten times for each benchmark, following the flow described in Section 3.A. This part of our study focuses on a quantitative analysis of the safe Vmin for diverse chips of the same architecture in order to expose the potential guardbands of each chip, as well as to quantify how the program behavior affects the guardband and to measure the core-to-core and chip-to-chip variation. For a significant number of benchmarks, we can see variations between different programs and different chips. Figure 3 represents the most robust core for each chip, and for these programs the Vmin varies from 885mV to 860mV for TTT, from 885mV to 870mV for TFF and from 900mV to 870mV for TSS. Considering that the nominal voltage for the X-Gene 2 is 980mV, there is a significant reduction of voltage without affecting the correct execution of programs, which is equal to at least 18.4% for the TTT and TFF chip, and 15.7% for the TSS chip. We also notice in Figure 5 that the workload-to-workload variation remains the same across the 3 chips of the same architecture; however, there is a relatively large variation among the chips. This means that there is a program dependency of Vmin behavior in all chips. Figure 6 shows the potential savings for the case that 8 different benchmarks run simultaneously: bwaves, cactusADM, dealll, gromacs, leslie3D, mcf, milc, namd. By exploiting the predictor's results, 12.8% power savings can be obtained by adjusting the voltage to the TTT Vmin without performance loss. Alternatively, the frequencies of the 2 weakest PMDs (0 and 1) can be reduced to 1.2 GHz (resulting in 25% performance loss) which will allow further reduction of the supply voltage to 885mV and energy savings up to 38.8%. Therefore, the predictor, apart from predicting the safe Vmin, it can also assist task scheduling in conjunction to frequency scaling according to the current workload on the system to further improve energy efficiency.

B. DRAM Characterization

In the setup used for the characterization of the DRAMs, we have separated the main memory into domains (based on the available channels) as shown on Figure 7, whose parameters can be set independently. This allowed us to isolate the critical kernel code and application data by placing them on a reliable memory domain to avoid any system crash that may occur under relaxed parameters. In this way, we have evaluated the energy-reliability trade-off of DRAMs under scaled supply voltage and refresh rate for a broad set of micro-benchmarks, HPC and Cloud workloads including Rodinia and Cloudsuite.

Our results show that the number of unique error locations, i.e. weak cells vary across the on board DRAM, chips and within the banks of each chip. Figure 8 shows the distribution of errors between 4 different DIMMs when running 4 parallel (8 threads) Rodinia benchmarks for 8

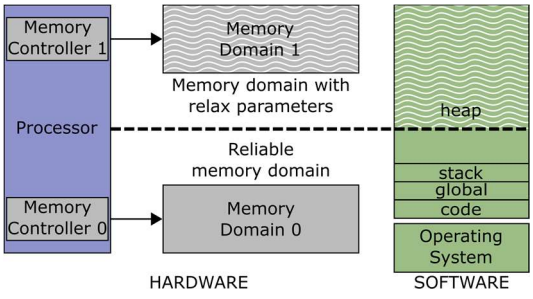

Figure 7: Experimental setup of heterogeneous reliability memory

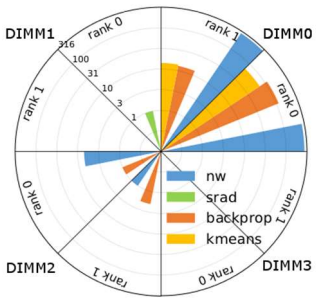

Figure 8: Distribution of errors across DIMMs and ranks

hours, with 8 Gb of input data. We observed that the majority of errors were triggered at DIMM0, while no errors have been reported for DIMM3. We also observe a large variation in the number of manifested errors across different applications: the Needleman-Wunsch benchmark triggered more than 600 errors, while only one error has been registered when we run the srad (Srad v2.0) benchmark.

Overall, our experiments indicated that by relaxing the refresh rate by 43x and voltage by 5% then the DRAM power usage can be reduced by 23.2% on average and by 12.3% on average when using the heterogeneous framework mentioned above for protecting the critical structures without compromising the reliability or performance of the executed applications. Interestingly, we observed that for a class of applications the number of errors could be reduced even under aggressively reduced refresh rate by ordering the memory access and ensuring that all accesses occur within a targeted time period that is less than the next scheduled refresh operation [14], a method which we can further exploit.

C. *Error-Resilient System Software*

System Software and especially the Hypervisor of UniServer must be resilient against memory and CPU errors. However, the overhead of resiliency should not outweigh the energy efficiency benefits achieved at EOP. A careful characterization of code and data structures is thus necessary to enable a selective and effective protection strategy.

We measured the Hypervisor memory footprint by repeatedly executing four instances of VMs, each of which accommodates a graph database benchmark (LDBC Social Network Benchmark [15] on top of Sparksee Graph Database). This application stresses the CPU, disk I/O and network. As shown in Figure 9, the Hypervisor footprint (red line) is always less than 7% compared to total utilized memory of the system. Similar observations hold for other applications we experimented with. This dictate placing the whole Hypervisor in reliable memory domain using the framework mentioned in Section 5.B can help ensure non-disruptive operation at low cost.

In order to characterize the sensitivity and significance of Hypervisor's internal data structures and code, we have applied fault injection to all statically allocated data structures of both Linux/KVM and QEMU [16]. Afterwards, for each execution we checked whether the data corruption resulted to a non-responsive Hypervisor, and marked this object accordingly as crucial or non-crucial for the Hypervisor state. Figure 10 and Figure 11 depict the results for the kernel- and the user-level components of the Hypervisor respectively. It is clear that there is a clustering in the criticality and sensitivity of

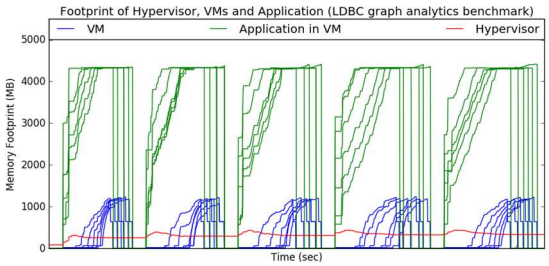

Figure 9: Memory footprint of Hypervisor, VMs and Application

data structures and code, according to their functionality. Those structures and code are varying for different workloads, however in a very predictable way. For example, for the results depicted in Figure 10 we experimented with a VM running a graph database benchmark mentioned above. Results show that data structures responsible for file system (fs), kernel, network (net) operations are evaluated as sensitive and should be protected. Those Linux/KVM modules are indeed the ones stressed by the specific application.

In order to improve the fault resilience of Linux/KVM itself, we also break their SMP assumptions and implement mechanisms that allow the migration of critical system operations to reliable cores. This significantly improved the observability of errors before they become fatal for the system, at a workload-dependent overhead of up to 7%.

D. *Total Cost of Ownership*

Figure 12 shows the savings in the TCO (y-right axis) as the voltage in different components is reduced and the probability for a failure (y-left axis), in terms of either application or system crash. As the graph shows, the TCO savings are increasing and reach the 3.6% point related to the nominal voltage setting. After that point, TCO savings are decreasing due to the higher probability of failure. Higher probability of failure causes resource overprovisioning to avoid violating the availability requirement of an application. As the figure shows even though energy savings are increasing monotonically, TCO does not have the same trends. This happens due to the extra cold spares that are needed due to the availability requirement violation of a specific application. Cold spares are server or component (DRAM, processor) modules needed for replacement when active servers or components failed. The fault rate of a server can be determined by the Mean Time To Failure of its components and the Mean Time To Repair. The cold spares are not active and only used when a server is down due to a failure. These spares are only accounted in the TCO with their capital expenses and not their operational expenses such as power. So, the point where the TCO savings are decreasing is the point that the cost of all the number of cold spares that are needed is overlapping the energy savings at a specific voltage setting.

When adopting the UniServer framework and reducing the CPU, SoC, or DRAM voltage while keeping a stable workload and CPU frequency, the probability of failure is moving closer to one, whereas energy savings are increasing. The lower the voltage, the less energy consumption; however, there may be more reliability issues. So, it depends also on the application requirements for finding the best voltage setting for executing the application.

Each application has its own requirements related to performance, power and reliability/availability. TCO can encapsulate all the metrics such as reliability (pfail) and energy in each voltage setting and can show the best voltage setting in which to run a specific application.

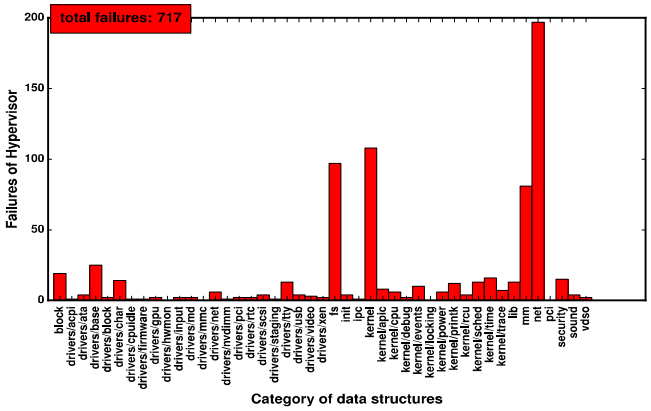

Figure 10: Sensitivity evaluation of structures of different kernel-level (Linux/KVM) Hypervisor components using fault injection

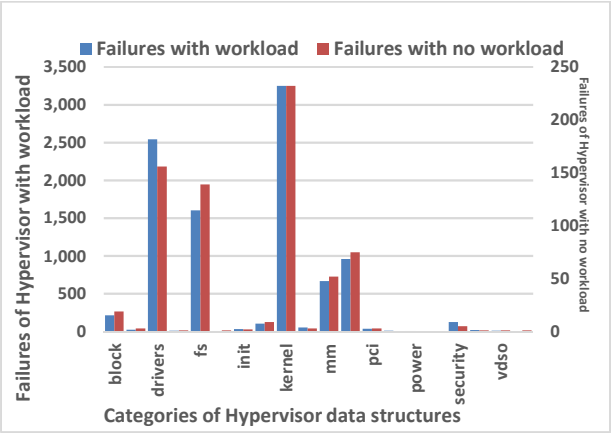

Figure 11: User-level Hypervisor (QEMU) fatal failures in case of errors in different structures

6. Enhancing the State-of-the-Art

Concerning the state-of the-art, a wealth of work exists on Dynamic Voltage and Frequency Scaling (DVFS) and variation aware design [3-9, 17-19], which UniServer enhances by developing automated procedures for characterizing, revealing and exploiting the true capabilities of each unit within commercial servers. The proposed approach provides a complete solution requiring minimum intrusion without any application level modifications. A handful of works have also suggested ways for enhancing the fault tolerance of system software [20-24]. One of the latest and most relevant works [24] require extensive Hypervisor modifications, which also are not applicable on KVM Hypervisor. The UniServer Hypervisor seeks resilience through a careful characterization of the criticality and sensitivity of Hypervisor data structures and code, and educated checking and selective checkpointing mechanisms, driven by this analysis. Other approaches, such as [25] and VMware vLockstep [26] achieve resilience by maintaining coherent replicas of VMs on different physical servers. Such approached may not be practical neither in Edge computing environments, where replication may not be possible, nor in power- or energy-constrained deployments targeted by UniServer. UniServer, also extends the state-of-the-art in resource management [27-30] by allowing data center resource manager to aggressively pursue power efficiency by enabling the operation of physical hosts in configurations beyond the conservative guard-bands currently imposed on the machines, by monitoring and predicting the node behaviour online.

7. Conclusions

This paper presents the basic ideas of the UniServer project which attempts to reduce hardware safety margins by utilizing representative stress cases, constant hardware monitoring and predictive mechanisms within commercial servers. The complete system stack approach includes a modified error-resilient Hypervisor and a cloud resource management software all being ported on a state of-the-art ARMv8 based microserver. Our results already indicate that extensive margins existing in the state of the art CPUs and DRAMs, while revealing the few kernel structures that are critical for maintaining non-disruptive system operation. In the next months of the project lifetime, we plan to enhance the developed technologies and demonstrate the gains by using smart emerging applications deployed in classical cloud business data-centers as well as in new environments closer to the data sources using the developed prototype. By doing so the developed prototype aspires to drive Edge computing and turn the opportunities in the emerging Big Data and IoT markets into real, smarter products.

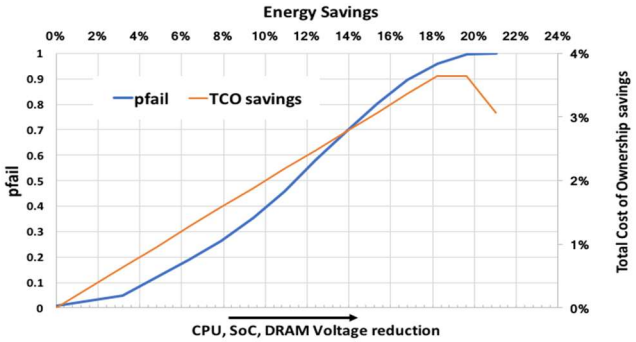

Figure 12: Energy savings across different error rates

References

[1] Cisco. Visual networking index: Global mobile data traffic forecast update 2014-2019.

[2] HP. White paper. http://h30507:www3:hp:com/t5/Cloud-Source-Blog/.

[3] K. A. Bowman, et al. "A 45 nm resilient microprocessor core for dynamic variation tolerance". IEEE JSSC, 2011.

[4] P. N. Whatmough, et al. "14.6 an all-digital power-delivery monitor for analysis of a 28nm dual-core arm cortex-a57 cluster", ISSCC 2015.

[5] V. J. Reddi et al. "Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling." MICRO, 2010.

[6] S. Borkar et al., "Parameter variations and impact on circuits and microarchitecture," DAC, 2003.

[7] H. Esmaeilzadeh et al. "Dark silicon and the end of multicore scaling." ISCA, 2011.

[8] G. Karakonstantis et al. "Containing the nanometer pandora-box: Cross-layer design techniques for variation aware low power systems," IEEE JETCAS, 2011.

[9] L. Leem et al. "Cross-layer error resilience for robust systems" IEEE ICCAD 2010.

[10] Y. Kim, et al. "AUDIT: Stress testing the automatic way", MICRO, 2012.

[11] OpenStack, "Open source software for creating private and public clouds," [Online]. Available: https://www.openstack.org/.

[12] J. L. Henning, "SPEC CPU2006 benchmark descriptions," SIGARCH Comput. Archit. September 2006.

[13] G. Papadimitriou et al., "Harnessing voltage margins for energy efficiency in multicore CPUs". MICRO 2017

[14] K. Tovletoglou et al., "Relaxing DRAM refresh rate through access pattern scheduling: A case study on stencil-based algorithms". IOLTS 2017

[15] O. Erling et al., "The LDBC Social Network Benchmark: Interactive Workload," ACM SIGMOD 2015.

[16] Fabrice Bellard. "QEMU, a fast and portable dynamic translator". *USENIX* ATEC, 2005.

[17] S. Das et al., "RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance," JSSCC 2009.

[18] D. M. Bull et al., "A Power-Efficient 32 bit ARM Processor Using Timing-Error Detection and Correction for Transient-Error Tolerance and Adaptation to PVT Variation" IEEE JSSC, 2011

[19] J. Liu et al., "RAIDR: Retention-aware intelligent DRAM refresh," ISCA, 2013.

[20] W. Gu, et al. "Characterization of Linux Kernel Behavior under Errors". IEEE DSN, 2003.

[21] T.-Y. Lee et al. "Fault isolation using stateless server model in L4 microkernel." ICCAE, 2010.

[22] A. Srivastava et al. "Efficient protection of kernel data structures via object partitioning." In Proceedings of the 28th annual Computer Security Applications Conference, 2012.

[23] F. David et al., "Building a self-healing operating system" DASC, 2007.

[24] X. Jin, et al. "FTXen: Making Hypervisor resilient to hardware faults on relaxed cores." IEEE HPCA, 2015.

[25] T. Bressoud, et al. "Hypervisor-based fault tolerance." ACM TOCS,, 1996.

[26] VMware (2009), 'VMware vSphere™ 4 Fault Tolerance: Architecture and Performance'

[27] A. Bahga et al., "Analyzing Massive Machine Maintenance Data in a Computing Cloud," IEEE TPDS, 2012.

[28] Daniel Dean, et al. "UBL: unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems". ICAC, 2012.

[29] P. Gaikwad et al., "Anomaly detection for scientific workflow applications on networked clouds". HPCS, 2016.

[30] Guan, Qiang, et al. "A Failure Detection and Prediction Mechanism for Enhancing Dependability of Data Centers," J CTE, 2012.