Ref. Ares(2018)1410165 - 14/03/2018

uniserver

D3.6 2nd Analysis of On-Chip Caches and Dynamic Memories

Contract number	688540
Project website	http://www.uniserver2020.eu
Contractual deadline	Project Month 24 (M24): 31st January 2018
Actual Delivery Date	???
Dissemination level	Public
Report Version	1.0
Main Authors	Konstantinos Tovletoglou (QUB), Lev Mukhanov (QUB), Georgios Karakonstantis (QUB), George Papadimitriou (UOA), Manolis Kaliorakis (UOA), Athanasios Chatzidimitriou (UOA), Dimitris Gizopoulos (UOA)
Contributors	
Reviewers	Arnau Prat(UPC)
Keywords	Characterization, on-chip caches, DRAMs, voltage/frequency/refresh rate scaling, corrected errors, uncorrected errors, silent data corruptions, crashes

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.

© 2018. UniServer Consortium Partners. All rights reserved

Disclaimer

This deliverable has been prepared by the responsible Work Package of the Project in accordance with the Consortium Agreement and the Grant Agreement Nr 688540. It solely reflects the opinion of the parties to such agreements on a collective basis in the context of the project and to the extent foreseen in such agreements.

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.

Confidentiality Note

This document may not be copied, reproduced, or modified in whole or in part for any purpose without written permission from the UniServer Consortium. In addition to such written permission to copy, reproduce, or modify this document in whole or part, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.

Change Log

Version	Description of change
0.1	Initial draft – Outline
0.2	Integrate
0.3	Changes in QUB part
0.4	Review from UoA
1.0	Final version for delivery to EC

© 2018. UniServer Consortium Partners. All rights reserved

Table of Contents

© 2017. UniServer Consortium Partners. All rights reserved

Index of Figures

© 2018. UniServer Consortium Partners. All rights reserved

Executive Summary

This document is a part of the Task T3.3 “Analysis of caches and dynamic memories” as presented in the Description of Action (DoA) of the UniServer project under Work Package 3 (WP3 – “Analysis and Enhancement of Hardware Substrate Outside Nominal Conditions”). This task uses the X-Gene 2 board as a target platform described in the Task 3.1 “Definition and enhancement of the target platform” and contributes to two objectives: (a) analysis of error behaviour for the processor cores and on-chip caches under various voltage and frequency settings and stress conditions, and (b) analysis of DRAM error behaviour operating under scaled refresh rates and supply voltage and various temperature conditions.

This deliverable focuses on the presentation of developed tools and methodologies to analyse error behaviour for the on-chip caches and dynamic memories of the target platform (X-Gene 2) operating under off-nominal Voltage/Frequency/Refresh rate (VFR) conditions. Making an extension to the characterization study discussed in D3.3, we identify workload-dependent parameters that affect reliability of the system operating under off-nominal conditions which can be exploited to build a failure prediction model (the task T4.4). The outcomes of this report are going to be used:

- To develop dedicated programs to stress the system operating under Voltage/Frequency/Refresh rate (VFR) conditions. These programs can be synthetic stress tests (“diagnostic virus”) for the cores, the on-chip caches and DRAM as presented in T3.2 and T3.3 of WP3 respectively and real application kernels as presented in D4.4 “Benchmark Suite for StressLog” which is a part of the task T4.3 in WP4.
- To provide valuable directions for building of the prediction model being developed in T4.4 and to be presented in D4.8.

1. Introduction

UniServer targets to improve performance and energy efficiency in cloud systems by utilizing the pessimistic Voltage/Frequency/Refresh rate (VFR) margins conventionally adopted by hardware vendors. However, system operation under off-nominal conditions increases the population of manifested errors compared to the operation under nominal conditions. Changing the supply voltage, frequency or the DRAM refresh rate may induce errors in different parts of the chip, i.e. within cores and memory subsystem. In depth characterization and analysis of system’s behavior under nominal and scaled VFR conditions is of major importance to finally provide a reliable, high performance and energy efficient system.

The insights that were revealed by this characterization step are valuable for many tasks and work packages of the UniServer project. They have guided decisions concerning the development of dedicated programs to stress the entire system in its operation limits under off-nominal VFR conditions. Moreover, the output of running all these stress programs under off-nominal conditions will feed the D4.8, with all the necessary insights about the failure model of caches and DRAM. The predictor module integrated with OS will be able to identify the best combination of VFR values in order to ensure the efficient operation of the system in terms of performance, energy and reliability (as presented in T4.4).

The goal of this deliverable is to present the results of the characterization for the on-chip cache memories of different levels, as well as off-chip DRAM when operating beyond the nominal conditions in the X-Gene 2 platform, extending the findings of the D3.3. The features of the targeted platform were described in detail in deliverable D3.1 “Definition of the UniServer Board”. Concerning the cores and the on-chip caches, a versatile automated framework was developed for system-level voltage and frequency scaling characterization of the Applied Micro’s X-Gene 2 micro-server family as was described in detail in D3.3. In summary, the framework provides fine-grained information about the system’s state by monitoring any abnormal behavior that may occur during under-scaled supply voltage and frequency conditions. In our preliminary results using our framework presented in D3.3, we observed significant variations when we scale the voltage and frequency running the same workload on different cores or running different workloads on the same core. We also discussed all the related work in the literature concerning the characterization of the caches and the impact of voltage noise in systems that operate under-off nominal voltage conditions. To extend our finding in this deliverable, we present the results of the characterization on the X-Gene 2 server with two new studies. In the first study, we use dedicated micro-viruses programs for fast characterization of

© 2018. UniServer Consortium Partners. All rights reserved

X-Gene 2 to reveal its safe operation margins, while in the second study we illustrate a statistical analysis study to predict the safe operation voltage margins for each core of the chip.

Concerning the characterization of Dynamic Random-Access Memory (DRAM), we extend the experimental framework on X-Gene 2 with a thermal testbed for conducting experiments under controlled DRAM temperature. This allows us to understand DRAM behavior under the temperature conditions which may vary in Cloud, but most importantly in Edge environments. In this deliverable, we analyze the number of manifested errors and correlate how different data and access patterns affect the error rate when DRAM operates under lowered supply voltage and relaxed refresh rates.

The document is organized in the following sections:

Section 2 presents the on-chip caches characterization. Using the characterization framework described in detail in D3.3, we present two new studies concerning the efficient development of cache and pipeline micro-viruses and a statistical analysis study that aims to predict the safe voltage operation limits of the ARMv8 cores under off-nominal voltage and frequency conditions.

Section 3 presents the characterization of dynamic memories: It presents the efforts of creating the experimental setups utilized during this period. Then the section presents the results of characterization for applications with different data sizes and frequency accesses patterns and the techniques used to correlate the number of errors for DRAM operating at the lowered supply voltage and relaxed refresh rates with parameters of the system. Continuing, this section introduces a thermal framework to control the DRAM temperature in order to understand the effects of data and frequency patterns. Lastly, we draw some conclusions and observations across our experiments

Finally, conclusions are drawn in Section4: It summarizes new contributions reported in this deliverable and discusses future work.

© 2018. UniServer Consortium Partners. All rights reserved

2. On-Chip Caches Characterization

The characterization results that were presented in D3.3 revealed important *core-to-core*, *benchmark-to-benchmark* and *chip-to-chip* variations. To further extend the characterization results that were presented in D3.3, in this deliverable we firstly illustrate the results of the characterization when three different chips (that were provided to UoA, UCY and QUB partners) operate in scaled voltage and refresh rate conditions, while they run the WSE DoSSensing application (Jammer Detector). Moreover, in this section we present two studies related to the characterization process of the X-Gene 2. The first study presents the development of micro-viruses that target all the levels of caches memory hierarchy (L1 Data and Instruction cache, unified L2 cache and L3 cache) as long as the pipeline, the ALUs and the FPUs of the ARMv8 cores of the X-Gene 2 chip in order to evaluate fast the safe operation limits of the chip. In the second study, we present a statistical analysis methodology to predict the safe voltage operation margins of the ARMv8 cores using as input the results from the characterization phase and the performance counters of the benchmarks during their entire execution.

2.1. Characterization of the WSE DoSSensing – Jammer Detector

To characterize the Jammer Detector that is described in detail in D7.4 “Evaluation of the Prototype and Comparison to State-of-the-Art in terms of Energy, Security and Determined Metrics of Success”, we used the framework described in detail in this deliverable that gives us the opportunity to change the PMD Voltage, the SoC Voltage, the DRAM Voltage and the DRAM refresh rate of the X-Gene 2 chips. Apart from the errors, the SDCs and the Crashes we also extended the framework in order to check the QoS requirements of the application.

To ensure the statistical significance of the delivered results, we launched 100 iterations of the characterization campaign in order to detect the safe operations margins. Moreover, we repeated the characterization experiments of the application in three different TTT chips of the three partners that are involved in the characterization of the X-Gene 2 (UoA, UCY, QUB).

We separated the characterization of the Jammer Detector in four different steps:

- Identification of the safe operation margins when we reduce the voltage of the PMDs.
- Identification of the safe operation margins when we reduce the voltage of the SoC.
- Identification of the safe operation margins when we reduce the voltage and we increase the refresh rate of the DRAM.
- Combination of the identified operation margins from the three previous steps concerning the PMDs and the DRAM in order to measure the maximum power savings for the 100 iterations.

For all the aforementioned steps, we ensure the correct operation and the QoS of the application. After identifying all the safe margins concerning the PMDs and the DRAM we measured the power savings in off-nominal conditions compared to the power that is consumed using the nominal values.

Table 1 presents the nominal values and the safe operations limits (best cases observed for 100 iterations of the experiment) that were identified for all the parameters and the three chips that were used for the evaluation of the Jammer Detector application.

Table 1: Safe operating limits for each of the chip (best cases for 100 iterations).

Parameter	Nominal	UoA chip	UCY chip	QUB chip
Vmin – **PMD**	980 mV	930 mV	910 mV	930 mV
Vmin – **SoC**	950 mV	920 mV	870 mV	900 mV
Vmin – **DRAM**	1500 mV	1428 mV	1428 mV	1428 mV
Refresh Rate min – **DRAM**	64 ms	2279 ms	2279 ms	2279 ms

© 2018. UniServer Consortium Partners. All rights reserved

Figure 1 illustrates the average power savings that were measured for the chip located in UoA for 100 iterations of the experiments. We can observe 20.2% power savings in total compared to the nominal values.

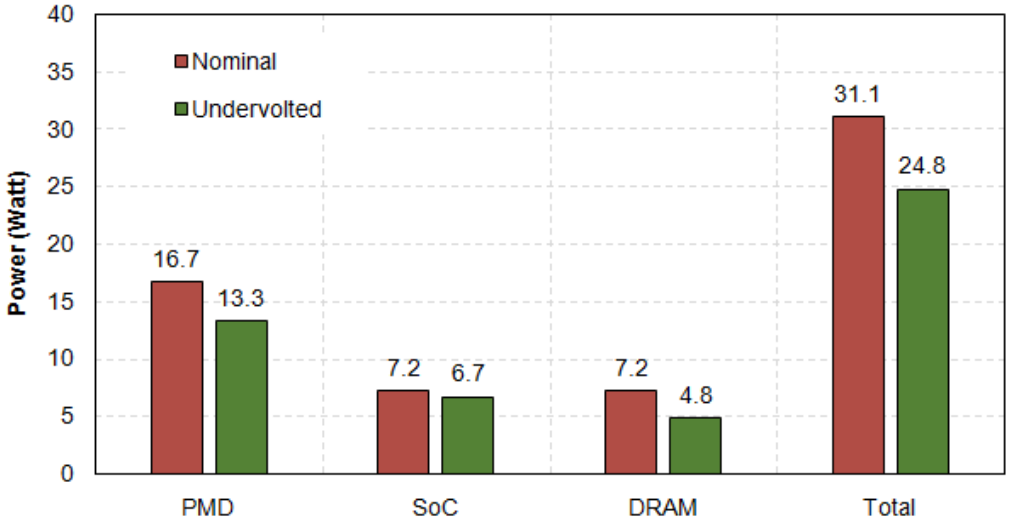

Figure 1: Average power savings for 100 iterations of the experiment in UoA chip.

Figure 2 presents the total power savings that were measured for the chip located in QUB. We can observe 12.3% power savings in total.

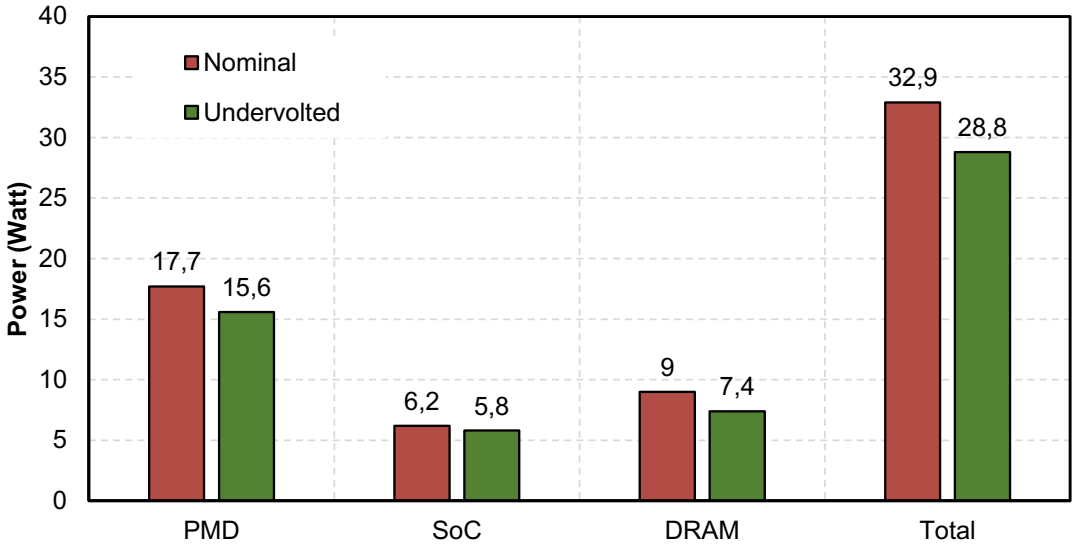

Figure 2:Average power savings for 100 iterations of the experiment in QUB chip.

Finally, Figure 3 illustrates the total power savings that were measured for the chip located in UCY. We can observe 12.5% power savings in total.

© 2018. UniServer Consortium Partners. All rights reserved

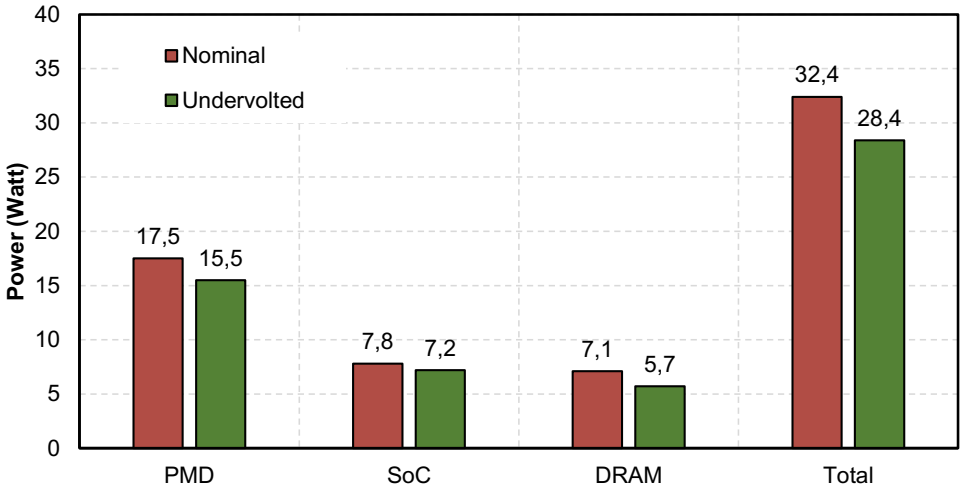

Figure 3: Average power savings for 100 iterations of the experiment in UCY chip.

2.2. Cache and Pipeline Micro-viruses

We developed diagnostic micro-viruses [1] to stress individually the fundamental microprocessor units (caches, ALU, FPU) that define the safe voltage margins of the micro-processor. We don't aim to reveal the absolute Vmin (which can be identified by worst-case voltage noise stress programs) since in X-Gene 2 there is no direct way for on-chip voltage noise measurements. However, we provide strong evidence (IPC and power measurements) that the micro-viruses are stressing the chips more intensively than the SPEC programs.

For the construction of the diagnostic micro-viruses we followed two different principles for the tests that target the caches and the pipeline, respectively. All diagnostic micro-viruses for the cases are small self-checking pieces of code. This means that the micro-viruses check if a read value is the expected one or not. There are previous studies for the construction of such tests, but they focus only on error detection (mainly for soft errors), and to our knowledge this is the first work that is performed in actual microprocessor chips; not in simulators or RTL level, which have no interference with the operating system and the challenges that it provides. We first present a brief overview of the challenges for the development of such system-level micro-viruses in a real hardware and the decisions we made in order to develop accurate self-checking tests for the caches and the pipeline.

Caches: For all levels of caches the first goal of the developed micro-viruses is to flip all the bits of each cache block from zero to one and vice versa. When the cache array is completely filled with the desired data, the micro-virus read iteratively all the cache blocks while the chip operates in reduced voltage conditions and identifies any corruptions of the written values, which cannot be detected by detection mechanisms of the cache, such as the parity protection that can detect only odd number of flips.

All caches in X-Gene 2 have pseudo-LRU replacement policy. All our micro-viruses focusing on any cache level need to "warm-up" the cache before the test begins, by iteratively accessing the desired data in order to ensure that all the ways of the cache are completely filled and accessed with the micro-viruses' desired patterns. We experimentally observed through the performance monitoring counters that the safe number of iterations that "warm-up" the cache with the desired data, before the checking phase begins, is log2(# of ways) to guarantee that the cache is filled only with the data of the diagnostic micro-virus.

In order to validate the entire cache array, it is important to perform write/read operations in all of the bit cells. For every single case, we allocate a memory chunk equal to the targeted cache size. As the storing of data is performed in cache block granularity, we need to make sure that our data storage is block-aligned, otherwise we will encounter undesirable block replacements that will not guarantee the complete utilization of the cache array. Assume for example that the first word of the physical frame will be placed at the middle of the cache block. This means that when the micro-virus fills the cache, practically, there will be half-block size words that will replace a desired previously fetched block in the cache. Thus, if the cache blocks are N, the

© 2018. UniServer Consortium Partners. All rights reserved

blocks that are written in cache will be N+1 (meaning one cache block will get replaced), and thus, the self-checking property may be jeopardized. To this end, for all cache-related micro-viruses we perform a check at the beginning of the test to guarantee that the allocated array is cache aligned (to be block aligned afterwards). Another factor that has to be considered in order to achieve full coverage of the cache array, is the cache coloring. Unless the memory is a fully associative one (which is not the case of the ARMv8 microprocessors), every store operation is indexed at one cache block depending on its address. For physically indexed memories, the physical address of the datum or instruction is used. However, because the physical addresses are not known or accessible from the software layer, special precautions need to be taken in order to avoid unnecessary replacements. To address this issue, we exploit a technique that is used to improve cache performance, known as cache coloring. If the indexing range of the memory is larger than the virtual page, 2 addresses with the same offset on different virtual pages are likely to conflict on the same cache block (due to the size of L1 caches (32KB), the bits that index the cache occur in page offset, and thus, there is no conflict; this is the case for L2 and L3 caches in our system). To avoid this situation, the indexing range is separated in regions equal to the page size, known as colors. It is then enough to use equal number of pages in each color to avoid conflicts. The easiest way to achieve this, is to allocate contiguous physical address range, which is possible at the kernel level using the kmalloc() call. The contiguous physical range will guarantee that all the data will be placed and fully occupy the cache, without replacements or unoccupied blocks.

Another challenge that the micro-viruses need to take into consideration is the interference of the branch predictors and the cache prefetchers. In our micro-viruses, the branch prediction mechanism (in particular the branch mispredictions that can flush the entire pipeline) may ruin the self-checking property of the micro-virus, by replacing or invalidating the necessary data or instruction patterns. Moreover, prefetching requests can modify the pre-defined access patterns of the micro-virus' execution. To eliminate these effects, the memory access patterns of the micro-viruses are modelled using the stride based model for each of the static load and stores of the micro-virus. Each of the static load and store in the workload walk a bounded array of memory references with a constant stride, greater than the X-Gene 2's prefetcher stride. In that way, the cache-related micro-viruses are executed without the interference of the branch predictor or the prefetcher. We validated this by leveraging the performance counters that measure the prefetch requests for L1 and L2 cache and the mispredictions, and no micro-virus counts any event in the related counters.

Pipeline: For the pipeline, we developed dedicated benchmarks that stress: (i) the Floating-Point Unit (FPU), (ii) the integer Arithmetic Logical Units (ALUs) and (iii) the entire pipeline using a combination of loads, stores, branches, arithmetic and floating-point unit operations. The goal is to trigger the critical paths that could possibly lead to an error during off-nominal operation voltage conditions.

Generally, for all micro-viruses, one primary thing that we need to take into consideration, is that due to the fact that micro-viruses are executed in the real hardware, with operating system, we need to isolate all the system's tasks to a single core. Assume for example that we run the L1 data or instruction micro-virus in Core 0. Each core has its own L1 cache, so we isolate all the system processes and interrupts in the Core 7, and we assign the micro-virus to Core 0. To do this we use the sched_setaffinity() call of the Linux kernel to set the process' affinity. In such a way, we ensure that only the micro-virus is executed in the desired core each time. We follow the same concept for all micro-viruses, except for L3 cache, because L3 is shared among all cores, so a small noise from the system processes is unavoidable.

We developed all diagnostic micro-viruses in C language (except for L1 Instruction cache micro-virus, which is ISA-dependent and is developed with a mix of C and ARMv8 assembly instructions). Moreover, the micro-viruses (except for L1 instruction cache's) check the microprocessor's parameters (cache size, # of cache ways, existence of prefetcher or not, page size, etc.) and adjust the micro-viruses code to the specific CPU. This way, the micro-viruses can be executed in any microarchitecture and can easily adapted to different ISAs. Note that the set of micro-viruses will be publically available.

Next, we discuss the implementation details of the micro-viruses for all cache levels (L1 Data Cache, L1 Instruction Cache, Unified L2 Cache, L3 Cache), the pipeline, the ALU and the FPU.

2.2.1. L1 Data Cache

For the first level data cache of each core, we allocated statically an array in memory with the same size as the L1 data cache. As the L1 data cache is no-write allocate, after the first write of the desired pattern in all

© 2018. UniServer Consortium Partners. All rights reserved

the words of the structure we need to read them first, in order to bring all the blocks in the first level of data cache. Otherwise, the blocks would remain in the L2 cache and we would have only write misses in the L2 cache. Moreover, due to the pseudo-LRU policy that is used in the L1 data cache, we read all the words of the cache three consecutive times (log2(# of ways) = log28=3) before the test begins, in order to ensure that all the blocks with the desired patterns are allocated in the first level data cache. With these steps, we achieve 100% read hit in the L1 data cache during the execution of the L1D micro-virus in undervolting conditions. The L1 Data micro-virus fills the L1 Data cache with three different patterns, each of which corresponds to a different micro-virus test. These tests are the all-zeros, the all-ones, and the checkerboard pattern. To enable the self-checking property of the micro-virus (correctness of execution is determined by the micro-virus itself and not externally), we check if each fetched word is equal to the expected value (the one stored before the test begins) at the end of the test.

2.2.2. L1 Instruction Cache

The concept behind the L1 Instruction Cache micro-virus is to flip all the bits of the instruction encoding in the cache block from zero to one and vice versa. In the ARMv8 ISA there is no single pair of instructions that can be employed to invert all 32 bits of an instruction word in the cache, so to achieve this we had to employ multiple instructions. The instructions listed in Table 2 are able to flip all the bits in the instruction cache from 0 to 1 and vice versa according to the Instruction Encoding Section of the ARMv8 manual.

Each cache block of the L1 instruction cache is able to hold 16 instructions because each instruction is 32-bit in ARMv8 and the L1 Instruction cache block size is 64 bytes. The size of each way of the L1 Instruction Cache is 32KB / 8 = 4KB, and thus, it is equal to the page size which is 4KB. As a result, there should be no conflict misses when accessing a code segment (see cache coloring previously mentioned) with size equal to the L1 Instruction cache (the same argument holds also for the L1 Data Cache).

The method we guarantee the self-checking property in the L1 Instruction cache micro-virus is the following: The L1 instruction cache array holds 8192 words (64 sets x 8 ways x 8 words in each cache block = 8192). We use 8177 words to hold the instructions of our diagnostic micro-virus, and the remaining 15 instructions (8177 + 15 = 8192) to compose the control logic of the self-checking property and the loop control. More specifically, we execute iteratively 8177 instructions and at the end of this block of code we expect the destination registers to hold a specific “signature” (the signature is the same for each iteration of the same group of instructions, but different among different executed instructions). If this “signature” is distorted then the micro-virus detects that an error occurred (for instance a bit flip in an immediate instruction resulted in the addition of a different value) and records the location of the faulty instruction as well as the expected and the faulty signature for further diagnosis. We iterate this code multiple times and after that we continue with the next block of code.

As in the L1 Data cache micro-virus, due to the pseudo-LRU policy that is used also in the L1 Instruction cache, we fetch all the instructions three consecutive times (log2(# of ways) = log28=3) before the test begins, in order to ensure that all the blocks with the desired instruction patterns are allocated in the L1 instruction cache. With these steps, we achieve 100% read hit in the cache (and thus cache stressing) during the undervolting campaign.

Table 2: ARMv8 Instructions used in the L1 Instruction micro-virus. The right column presents the encoding of each instruction in bit granularity to demonstrate that all the bits in the cache block get flipped.

Instruction	Encoding
add x28, x28, #0x1	1001 0001 00 0000 0000 0001 11100 11100
sub x3, x3, #0xffe	1101 0001 00 1111 1111 1110 00011 00011
madd x28, x28, x27, x27	1001 1011 00 0110 1101 1011 11100 11100
add x28, x28, x27, asr #2	1000 1011 10 0110 1100 0010 11100 11100
add w28, w28,w27,lsr #2	0000 1011 01 0110 1100 0010 11100 11100
nop	1101 0101 00 0000 1100 1000 00000 11111
bics x28, x28, x27	1110 1010 00 1110 1100 0000 11100 11100

© 2018. UniServer Consortium Partners. All rights reserved

2.2.3. Unified L2 Cache

The L2 cache is a 32-way associative PIPT cache with 128 sets; thus, the bits of the physical address that determine the block placement in the L2 cache are bits [12:6] (as shown in Figure 4). Moreover, the page size we rely on is 4KB and consequently the page offset consists of the 12 less significant bits of the physical address. Accordingly, the most significant bit (bit 12) of the set index (the dotted square in Figure 4) is not a part of the page offset. If this bit is equal to 1, then the block is placed in any set of the upper half of the cache, and in the same manner, if this bit is equal to 0, the block is placed in a set of the lower half of the cache. Bits [11:6] which are part of page/frame offset determine all the available sets for each individual half.

In order to guarantee the maximum block coverage (meaning to completely fill the L2 cache array), and thus to fully stress the cache array, the L2 micro-virus should not depend on the MMU translations that may result in increased conflict misses. The way to achieve this is by allocating memory that is not only virtually contiguous (as with the standard C memory allocation functions used in user space), but also physically contiguous by using the kmalloc() function (see cache coloring previously mentioned). The kmalloc() function operates similarly to that of user-space's familiar memory allocation functions, with the main difference that the region of physical memory allocated by kmalloc() is physically contiguous. This guarantees that in one half of the allocated physical pages, the most significant bits of their set index are equal to one and the other half are equal to zero.

Given that the replacement policy of the L2 cache is also pseudo-LRU, the L2 micro-virus needs to iteratively access five times the allocated data array (log2(# of ways) = log232=5), to ensure that all the ways of each set contain the correct pattern. Furthermore, due to the fact that the L1 data cache has write-through policy and the L2 cache has write allocate policy, the stored data will reside in the L2 cache right after the initial writes (there are no write backs). Another requirement for the L2 micro-virus is that it should access the data only from the L2 cache during the test and not from the L1 data cache, to completely stress the former one. We satisfy this requirement by using a stride access scheme for the array with a block stride of one block (8 words stride). Therefore, in the first iteration the L2 micro-virus accesses the first word of each block, in the second iteration it accesses the second word of each block, and so on. Thus, it always misses the L1 Data cache. By accessing the data using these strides, the L2 micro-virus also overcomes the prefetching requests. Note that the L1 instruction cache can completely hold all the L2 diagnostic micro-virus instructions, so the L2 cache holds only the data of our test.

To verify the above, we isolated all the system processes by forcing them to run in different cores from the one that executes the L2 diagnostic micro-virus, by setting the system processes' CPU affinity and interrupts to a different core, and we measured the L1 and L2 accesses and misses after we have already "trained" the pseudo-LRU with the initial accesses. We measure these micro-architectural events by leveraging the built-in performance counters. The performance counters show that the L2 diagnostic micro-virus always misses the L1 Data cache and always hits the L1 Instruction cache, and it hits the L2 cache in the majority of accesses. Specifically, the L2 cache has 4096 blocks and the maximum number of block misses we observed was 32 at most for each execution of the test (meaning 99.2% coverage). In such a way, we verify that the L2 micro-virus completely fills the L2 cache. The L2 micro-virus fills the L2 cache with three different patterns, each of which corresponds to a different micro-virus test. These tests are the all-zeros, the all-ones, and the checkerboard pattern. To enable the self-checking property into this micro-virus, at the end of the test we check if each fetched word is equal to the expected value (the one stored before the test begins).

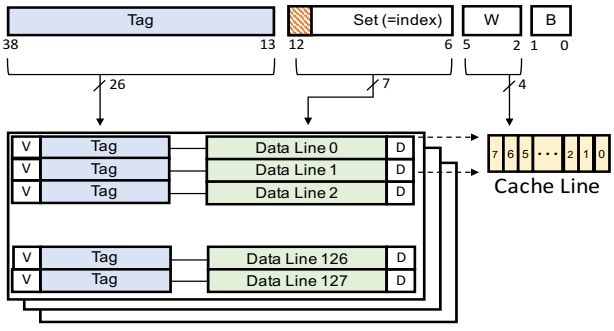

Figure 4: A 256KB 32-way set associative L2 cache.

© 2018. UniServer Consortium Partners. All rights reserved

2.2.4. L3 Cache

The L3 cache is a 32-way associative PIPT cache with 4096 sets and is organized in 32 banks; so, each bank has 128 sets and 32 ways. Moreover, the bits of the physical address that determine the block placement in the L3 cache are the bits [12:6] (for choosing the set in a particular bank) and the bits [19:15] for choosing the correct bank. Based on the above, in order to fill the L3 cache we allocate physically contiguous memory with kmalloc(). However, kmalloc() has an upper limit of 128 KB in older Linux kernels and 4MB in newer kernels (like the one we are using; we use CentOS 7.3 with Linux kernel 4.3). This upper limit is a function of the page size and the number of buddy system freelists (MAX_ORDER). The workaround to this problem is to allocate two arrays with two calls to kmalloc() and each array's size should be half the size of the 8M L3 cache. The reason that this workaround will result in full block coverage in the L3 cache is that 4MB chunks of physically contiguous memory gives us contiguously the 22 less significant bits, while we need contiguously only the 20 less significant (for the set index and the bank index). Moreover, we should highlight that the L3 cache behaves as a non-inclusive victim cache. In response to an L2 cache miss from one of the PMDs, agents forward data directly to the L2 cache of the requestor, bypassing the L3 cache. Afterwards, if the corresponding fill replaces a block in the L2 cache, a write-back request is issued, and the evicted block is allocated into the L3 cache. On a request that hits the L3 cache, the L3 cache forwards the data and invalidates its copy, freeing up space for future evictions. Since data may be forwarded directly from any L2 cache, without passing through the L3 cache, the behavior of the L3 cache increases the effective caching capacity in the system.

Due to the pseudo-LRU policy, as in the L2, the L3 micro-virus is designed accordingly to perform five sequential writes (log2(# of ways) = log232=5) to cover all the ways before the test begins, and the read operations afterwards are performed by stride of 1 block (to bypass the L2 cache and the prefetcher, so the micro-virus only hits the L3 cache and always misses the L1 and L2 caches). The L3 diagnostic micro-virus fills the L3 cache with three different patterns, each of which corresponds to a different micro-virus test. These tests are the all-zeros, the all-ones, and the checkerboard pattern. To enable the self-checking property, at the end of the test we check if each fetched word is equal to the expected value (the one stored before the test begins).

However, in contrast to the L2 diagnostic micro-virus, in the L3 micro-virus we do not have the necessary tools to prove the complete coverage of the L3 cache due to the fact that there are no built-in performance counters in X-Gene 2 that report the L3 accesses and misses. However, by using the events that correspond to the L1 and L2 accesses, misses and write backs we check that all the requests from the L3 micro-virus miss the L1 and L2 caches, and thus only hit the L3 cache. Finally, we should highlight that the shared nature of the L3 cache forced us to try to minimize the number of the running daemons in the system in order to reduce the noise in the L3 cache from their access to it.

2.2.5. ALU

X-Gene 2 features a 4-wide out-of-order superscalar microarchitecture. It has one integer scheduler and two different integer pipelines: a Simple Integer pipeline, and a Simple+Complex Integer pipeline. The integer scheduler can issue two integer operations per cycle; each of the other schedulers can issue one operation per cycle (the integer scheduler can issue 2 simple integer operations per cycles; for instance, 2 additions, or 1 simple and 1 complex integer operation; for instance, 1 multiplication and 1 addition).

The execution units are fully pipelined for all operations, including multiplications and multiply-add instructions. ALU operations are single-cycle. The fetch stage can bring up to 16 instructions (same size as a cache block) per cycle. The fetch attempts to always bring up to 16 instructions from the same cache block or by two adjacent cache blocks. If the fetch begins in the middle of a cache block (unaligned), the next cache block will also be fetched in order to have 16 instructions available for further processing, and thus there will be a block replacement on the Instruction Buffer. To this end, we use NOP instructions to ensure that the first instruction of the execution block is block aligned, so that the whole cache block is located to the instruction buffer each time. For the above microarchitecture, we developed the ALU self-testing micro-virus, which avoids data and control hazards and iterates 1000 times over a block of 16 instructions (that resides in the Instruction buffer, and thus the L1 instruction and data cache are not involved in the stress testing process). After completing 1000 iterations, it checks the value of the registers involved in the calculations by comparing them with the expected values. After reinitializing the values of the registers, we repeat the same test 70M times, which is approximately 60 seconds of total execution (of course, the number of executions

© 2018. UniServer Consortium Partners. All rights reserved

and the overall time can be adjusted). Therefore, we execute code that resides in the instruction buffer for 1000 iterations of our loop and then we execute code that resides in 1 block of the cache after the end of these 1000 iterations. Because the instructions are issued and categorized in groups of 4 (X-Gene 2 issues 4 instructions), and the integer scheduler can issue 2 of them per cycle we can't achieve the theoretical optimal IPC of 4 Instructions per Cycle only with Integer Operations. Furthermore, we try to have in each group of 4 instructions, instructions that stress all the units of all the issue queues like the adder, the shifter and multiplier. Specifically, the ALU micro-virus consists of 94% integer operations and 6% branches.

2.2.6. FPU

Aiming to heavily stress and diagnose the FPU, we perform a mix of diverse floating-point operations, by avoiding data hazards (thus stalls) among the instructions and using different inputs to test as many bits and combinations as possible. To implement the self-checking property of the micro-virus, we execute the floating-point operations twice, with the same input registers and different result registers. If the destination registers of these two same operations have different result, our self-test notifies that an error occurred during the computations. For every iteration, the values of the registers (for all of the FPU operations) are increased by a non-fixed stride that is based on the calculations that take place. The values in the registers of each loop are distinct between them and between every loop. Moreover, we ensure that the first instruction of the execution block is cache aligned (as in the ALU micro-virus), so the whole cache block is located to the instruction buffer each time.

2.2.7. Pipeline

Apart from the dedicated benchmarks that stress independently the ALU and the FPU, we have also constructed a micro-virus to stresses simultaneously all the issue queues of the pipeline. Between two consecutive "heavy" (high activity) floating-point instructions of the FPU test (like the consecutive *multply add*, or the *fsqrt* which follows the *fdiv*) we add a small iteration over 24 array elements of an integer array and a floating-point array. To this end, during these iterations, the "costly" instructions such as multiply add have more than enough cycles to calculate their result, while at the same time we perform load, store, integer multiplication, exclusive or, subtractions and branches. All instructions and data of this micro-virus are located in L1 cache in order to fetch them at the same cycle to avoid high cache access latency. As a result, the "pipeline" micro-virus has a great variety of instructions which stress in parallel all integer and FP units. This micro-virus consists of 65% integer operations and 23.1% floating point operations, while the rest 11.9% are branches.

2.2.8. Micro-viruses validation

In the previous section, we described the challenges and our solutions to the complex development process of the micro-viruses and how we verified their coverage using the machine performance monitoring counters. However, it is essential to validate the stress and utilization of the micro-viruses on the microprocessor. To this end, we measure the IPC and power consumption for both micro-viruses and SPEC CPU2006 benchmarks. Note that the micro-viruses were neither developed to provide power measurements nor performance measurements. We present the IPC and power consumption measurements of the micro-viruses only to verify that they sufficiently stress the targeted units. IPC and power consumption along with the data footprints of the micro-viruses (complete coverage of the caches bit arrays) are highly accurate indicators of the activity and utilization of a workload on a microprocessor. Figure 5 and Figure 6 present the IPC and power consumption measurements, respectively, for both the micro-viruses and the SPEC CPU2006 benchmarks.

As shown in these figures, the proposed micro-viruses for fast voltage margins identification provide very high IPC compared to most SPEC benchmarks on the target X-Gene 2 CPU. In addition, we assessed the power consumption using the dedicated power sensors of the X-Gene 2 microprocessor (located in standby power domain), to take accurate results for each workload. We performed measurements for two different voltage values; at the nominal voltage (980mV) and at 920mV, which is a voltage step that all of the micro-viruses and benchmarks can be reliably executed (without Silent Data Corruptions (SDCs), errors or crashes). Figure **6** shows that the maximum and average power consumption of the micro-viruses are comparable to the SPEC CPU2006. In the same figure, we can also see the differences concerning the energy efficiency when operating below nominal voltage conditions, and this can strengthen the need for identifying the pessimistic voltage margins of a microprocessor. As we can see, in the multi-core execution

© 2018. UniServer Consortium Partners. All rights reserved

we can achieve 12.6% energy savings (considering that the maximum TDP of X-Gene 2 is 35W), by reducing the voltage 6.2% below nominal, where all of the three chips operate reliably.

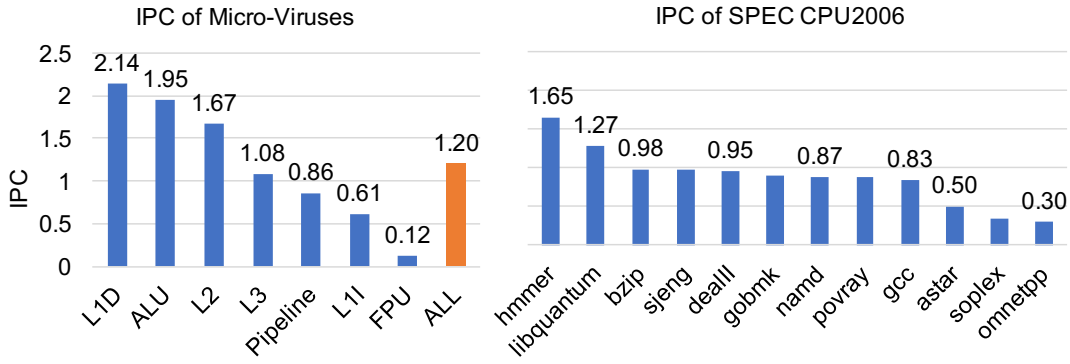

Figure 5: IPC measurements for both micro-viruses (top) and SPEC CPU2006 benchmarks (bottom).

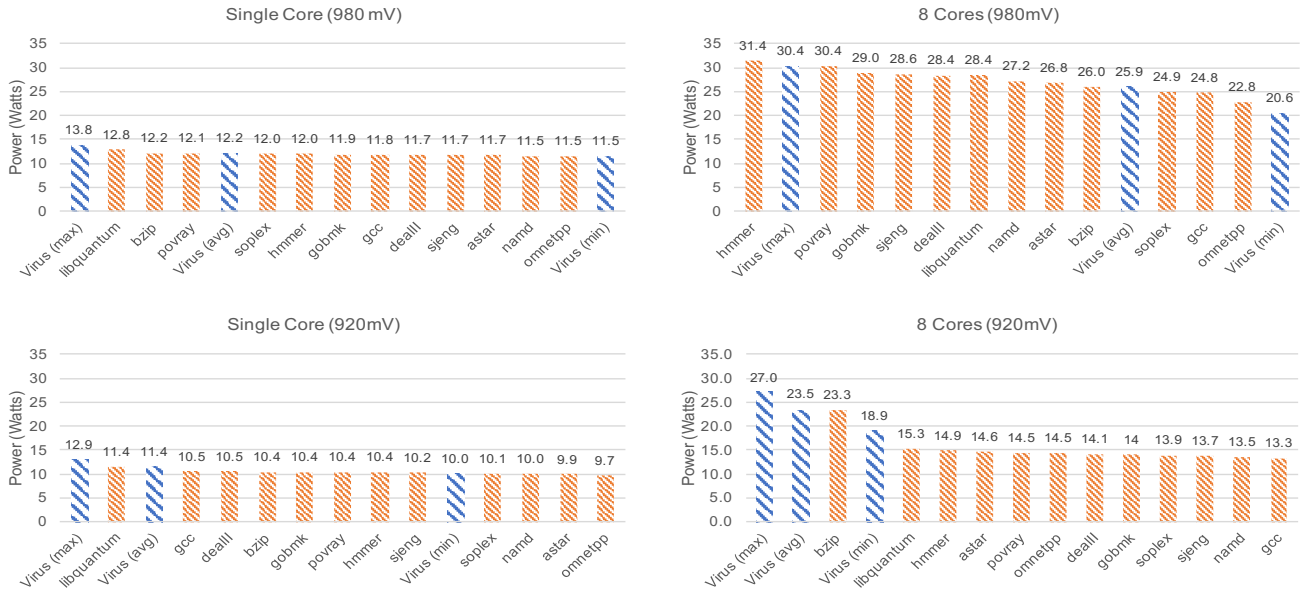

Figure 6: Power consumption measurements for both the micro-viruses and the SPEC CPU2006 benchmarks. The graphs at the top show the power consumption at nominal voltage (980 mV), when running on one core (left) and on all 8 cores concurrently (right). The bottom graphs show the power measurements when the microprocessor operates at 920mV, in order to present the energy efficiency when operating below nominal voltage conditions.

2.2.9. Experimental Evaluation

For the evaluation of the micro-viruses' ability to reveal the Vmin of X-Gene 2 CPU chips and their cores, we used three different chips: TTT, TFF, and TSS from the AppliedMicro's (APM) X-Gene 2 micro-server family.

Using the I2C controller we decrease the voltage of the domains of the PMDs and the SoC at 5mV steps, until the lowest voltage point (safe Vmin) before the occurrence of any error (corrected and uncorrected – reported by the hardware ECC), SDC (Silent Data Corruption – output mismatch) or Crash. Due to the non-deterministic behavior of a real machine (all of our experiments were performed on the actual X-Gene 2 chip), we repeat each experiment 10 times and we select the execution with the highest safe Vmin (the worst-case scenario) to compare with the micro-viruses.

We experimentally obtained also the safe Vmin values of the 12 SPEC CPU2006 benchmarks on the three X-Gene 2 chips (TTT, TFF, TSS), running the entire time-consuming undervolting experiment 10 times for each benchmark. These experiments were performed during a period of 2 months on a single X-Gene 2 machine. We also ran our diagnostic micro-viruses, with the same setup for the 3 different chips, as for the SPEC CPU2006 benchmarks. This part of our study focuses on:

© 2018. UniServer Consortium Partners. All rights reserved

1. the quantitative analysis of the safe Vmin for three significantly different chips of the same architecture to expose the potential guard-bands of each chip,
2. the demonstration of the value of our diagnostic micro-viruses which can stress the individual components, and reveal virtually the same voltage guardbands compared to benchmarks.

The voltage guardband for each program (benchmark or micro-virus) is defined as the safest voltage margin between the nominal voltage of the microprocessor and its safe Vmin (where no ECC errors or any other abnormal behavior occur).

As we discussed earlier, to expose these voltage margins among cores in the same chip and among the three different chips by using the 12 SPEC CPU2006 benchmarks, we needed ~2 months for each chip. On the contrary, the same experimentation by using the micro-viruses needs ~3 days, which can expose the corresponding safe Vmin for each core. In Figure 7 and Figure 8 we notice that the micro-viruses provide the same or higher Vmin than the benchmarks for 19 of the 24 cores (3 chips times 8 cores). There are a few cases that benchmarks have higher Vmin in 5 cores (the difference between them is at most 5mV – 0.5%) but in orders of magnitude shorter time. Such differences (5mV or even higher) can occur even among consecutive runs of the same program, in the same voltage due to the non-deterministic behavior of the actual hardware chip. This is why we run the benchmarks 10 times and present only the maximum safest Vmin. For a significant number of programs (benchmarks and micro-viruses), we can see variations among different cores and different chips. Figure 7 and Figure 8 represent the maxi-mum safe Vmin for each core and chip among all the benchmarks (blue line) and all micro-viruses (orange line). Considering that the nominal voltage in PMD voltage domain (where these experiments are executed) is 980mV, we can observe that the Vmin values of the micro-viruses are very close to the corresponding safe Vmin provided by benchmarks, but in most cases higher. The core-to-core and chip-to-chip relative variation among the three chips are also revealed with the micro-viruses. Both the SPEC CPU2006 benchmarks and the micro-viruses provide similar observations for core-to-core and chip- to-chip variation. For instance, in TTT and TFF chip, cores 4 and 5 are the most robust cores. This property holds in the majority of programs but can be revealed by the micro-viruses in several orders of magnitude shorter characterization time.

At the bottom-right diagram of Figure 8, we show the undervolting campaign in the SoC voltage domain (which is the focus of the L3 cache micro-virus). In X-Gene 2 there are 2 different voltage domains: the PMD and the SoC. The SoC voltage domain includes the L3 cache. Therefore, this figure presents the comparison of the L3 diagnostic micro-virus with the 12 SPEC CPU 2006 benchmarks that were executed simultaneously in all 8 cores (8 copies of the same benchmark) by reducing the voltage only in the SoC voltage domain. In this figure, we also notice that in TTT/TFF the difference of Vmin between the benchmark with the maxi-mum Vmin and the self-test is 5mV, while in TSS the micro-viruses reveal the Vmin at 20mV higher than the benchmarks. Note that the nominal voltage for the SoC domain is 950mV.

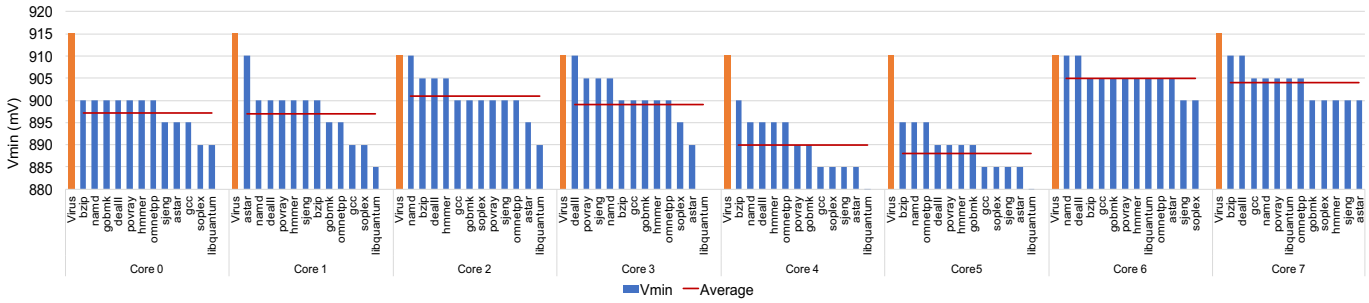

Figure 7: Detailed comparison of V_{min} between the 12 SPEC CPU2006 benchmarks and micro-viruses for the TSS chip.

© 2018. UniServer Consortium Partners. All rights reserved

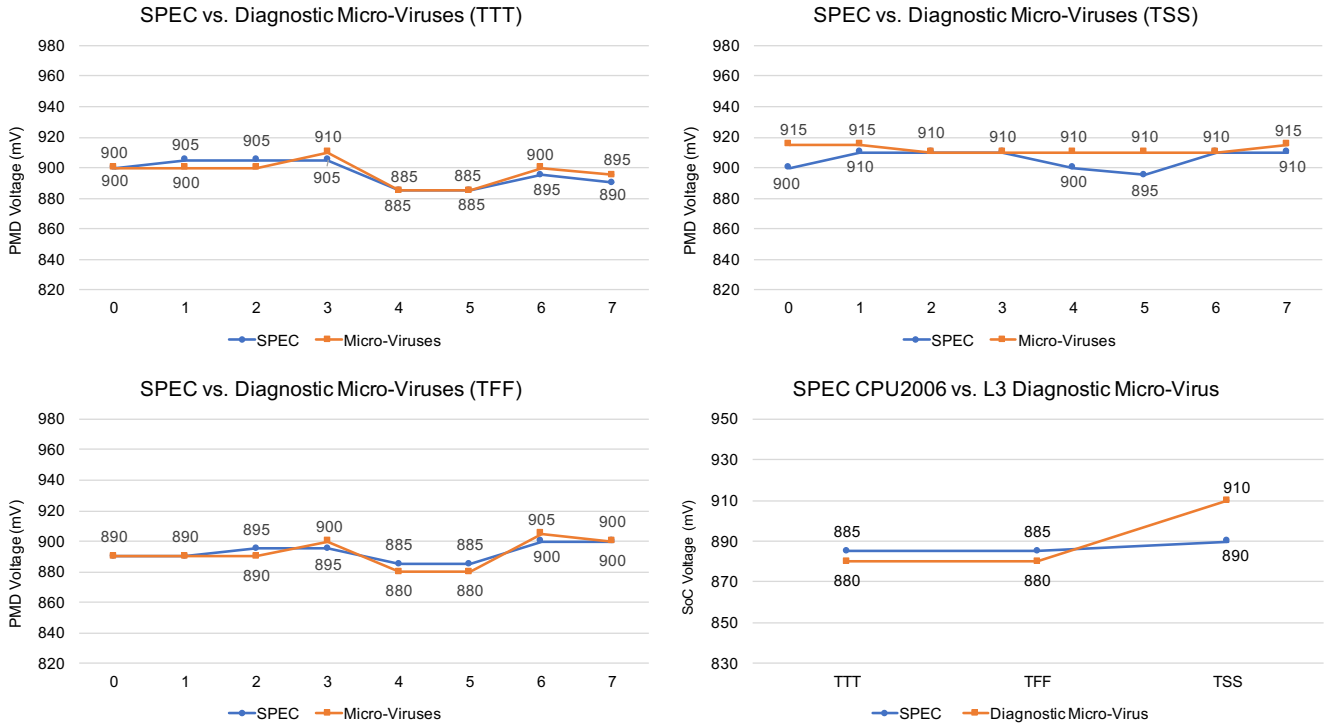

Figure 8: Maximum V_{min} among 12 SPEC CPU2z06 benchmarks and the proposed micro-viruses for TTT, TSS and TFF in PMD domain. Rightmost at bottom shown the maximum V_{min} of 12 SPEC CPU2006 benchmarks and the proposed L3 micro-virus in SoC domain.

By using the proposed micro-viruses, we can detect very accurately (divergences have short range, at most 5mV) the safe voltage margins for each chip and core, instead of running time-consuming benchmarks. According to our experimental study, the micro-viruses reveal higher Vmin (meaning lower voltage margin) in the majority of cores in the three chips we used. Specifically, 19 out of 24 in total cores, the micro-viruses expose higher or the same safe Vmin compared to the SPEC CPU2006 benchmarks. For the specific ARMv8 design, we point and discuss the core-to-core and chip-to-chip variation, which are important to reduce the power consumption of the microprocessor.

Core-to-Core Variation: There are significant divergences among the cores due to process variation. Process variation can affect transistor dimensions (length, width, oxide thick- ness etc.) which have direct impact on the threshold voltage of a MOS device, and thus, on the guard-band of each core. We demonstrate that although micro-viruses can reveal similar divergences as the benchmarks among the different cores and chips, however, in most of the cases, micro-viruses expose lower divergences among cores in contrast to time consuming SPEC CPU2006 benchmarks. As shown in Figure 7, our micro-viruses reveal higher safe Vmin for all the cores than the benchmarks, and also, we notice that the workload-to-workload differences are very high (up to 30mV). Therefore, due to the diversity of code execution of benchmarks, it is difficult to choose one benchmark that provides the highest Vmin. Different benchmarks provide significantly different Vmin at different cores in different chips. Therefore, it is necessary excessively large number of different benchmarks to have a safe result concerning the voltage margins identification. Thanks to micro-viruses, which fully stress the fundamental units of the microprocessor, cores' guard-bands can be safely determined (regarding the safe Vmin) at a very short time, and guide energy efficiency when running typical applications.

Chip-to-Chip Variation: As Figure 8 presents for the TTT and TFF chips, PMD 2 (cores 4 and 5) is the most robust PMD for all three chips (can tolerate up to 3.6% more undervolting compared to the most sensitive cores). We can notice that (on average among all cores of the same chip) the TFF chip has lower Vmin points than the TTT chip, in contrast to TSS, which has higher Vmin points than the other two chips, and thus, can deliver smaller power savings.
Diagnosis: By using the diagnostic micro-viruses we can also determine if and where an error or a silent data corruption (SDC) occurred. Through this component-focused stress process we have observed the following:

1. SDCs occur when the pipeline gets stressed (ALU, FPU and Pipeline tests), and

© 2018. UniServer Consortium Partners. All rights reserved

2. the cache bit-cells operate safely at higher voltages (the caches tests crash lower than the ALU and FPU tests).

Both observations show that the X-Gene 2 is more susceptible to timing-path failures than to SRAM array failures [2]. Previous studies on Intel Itanium have shown a large region of voltage values that contains only ECC corrected errors during undervolting. By reducing the voltage on those chips, the number of ECC corrected errors increases gradually for quite many voltage steps until it exposes the first abnormal behavior (SDC/crash). Unlike these studies, a major finding of our analysis using the micro-viruses for ARMv8-compliant multicore CPUs is that SDCs (derived from pipeline stressing using the ALU, FPU and Pipeline micro-viruses) appear at higher voltage levels than corrected errors when cache arrays get stressed by cache-related micro-viruses. We believe that the reason is that unlike other server-based CPUs (like Itanium), X-Gene 2 does not deploy circuit-level techniques (Itanium performs continuous clock-path de-skewing during dynamic operation [3]), and thereby, when the pipeline gets stressed, X-Gene 2 produces SDCs due to timing-path failures

2.3. Statistical Analysis based on Characterization Results

To understand the behavior of the X-Gene 2 chip, we used the output from the characterization phase in a statistical analysis scheme [4]. Statistical analysis methods are appealing to predict the safe operational margins at the system level as they do not induce area overheads and they can be applied during manufacturing or after the chips' release to the market. Thus, we launched a comprehensive statistical analysis of the behavior of ARMv8 64-bit cores that are part of the enterprise 8-core X-Gene 2 micro-server family when they operate in scaled voltage conditions. Our prediction schemes that use real hardware counters as input are based on linear regression models with several feature selection techniques that aim to predict the safe voltage margins of any given workload when the cores operate in scaled conditions. Our findings show that the power savings of operation above the safe V_{min} can be up to 17.52% compared to operation at nominal voltage or up to 20.28% when we use a more aggressive scheme that targets the Severity metric (operation in unsafe regions beyond V_{min}).

For our analysis, we used the linear regression model that can be easily calculated on hardware in contrast to the more complex non-linear models, as it is able to provide high accuracy with relatively small population of independent variables. The regression techniques calculate a function to predict a value of the dependent variable from a set of independent variables that are called *features*. Assuming a set of $x_1, x_2, x_3, \ldots, x_N$ independent variables and y the dependent variable, our analysis is based on the Ordinary Least Squares (OLS) model that calculates a set of weights β (one for each variable x) and an error term e, according to the formula:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + e_i \qquad (1)$$

In eq. (1), y_i is the i^{th} response value (i.e. V_{min} or Severity), x_{ji} is the j^{th} feature (e.g. hardware counter values) evaluated at the i^{th} observation, and e_i is the i^{th} statistical error. The goal of the model is to deliver the optimal values of the coefficients $\beta_1, \beta_2, \beta_3, \ldots, \beta_k$ so as to minimize the sum of the squares of the differences between the predicted and the observed responses of the test dataset.

We used as inputs for our models the output from the characterization phase (V_{min} and Severity values) and the performance counters from the entire execution of the workloads in nominal voltage conditions using the *perf* tool. For the implementation of all our models, we used the *sklearn* python libraries [5].

The X-Gene 2 provides 101 hardware counters that report all the major events concerning the memory hierarchy (accesses and misses in all cache levels, TLB and page walks levels, prefetches, etc.), the core pipeline (flushes, mispredictions, etc.) and the system (bus accesses, etc.). Before training the linear regression model and in order to eliminate and select the most appropriate counters for it, we used three different *feature selection techniques*:

1) F_regression: This approach is based on F-distribution and computes the correlation of each independent feature X[:, i] with the dependent variable y, according to the following formula:

$$\rho_i = \frac{\big(X[:,i] - mean(X[:,i])\big) * (y - mean(y))}{std(X[:,i]) * std(y)} \qquad (2)$$

© 2018. UniServer Consortium Partners. All rights reserved

Then, for each feature, the *F-value* is computed according to eq. (3), where *n* is the population of *y* value. Finally, using the *F-value*, the associated *p-value* of each feature is determined [6].

$$F_i = \frac{{\rho_i}^2}{1 - {\rho_i}^2} * (n - 1) \qquad (3)$$

Subsequently, the *F-values* and *p-values* of all features are sorted and finally, the best *k* features with the best values are selected and used as input from the linear regression model. Note that the lower the *p-value* and the higher the *F-value*, the more efficient the model becomes when we include this feature in the model.

2) **Recursive Feature Elimination (*RFE*)**: The goal of *RFE* is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features, and weights are assigned to them according to their correlation coefficient. Then, the least important features are pruned from the current set of features and new weights are assigned to the remaining features. This procedure is recursively repeated on the pruned set until the desired number of features is reached.

3) **Polynomial Feature Transformation (*Polynomial*)**: To evaluate the correlation between two or more features, we implemented polynomial feature transformation that generates a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to a specified degree. For example, if an input sample of features is two-dimensional and consists of two features [x_1, x_2], the new polynomial feature matrix of degree-2 will be [1, x_1, x_2, x_1^2, x_1x_2, x_2^2]. In this study, we implemented the polynomial feature transformation for all the cases targeting either the V_{min} or the Severity with a degree of correlation equal to 2. Our experiments revealed that a degree greater than 2 cannot provide better accuracy to our prediction models. After this transformation, we use either *f_regression* or *RFE* feature selection to select the best features for our linear regression models.

The evaluation of our models' accuracy targeting either the V_{min} or the Severity, was done by: (a) using the coefficient of determination (R^2) that assesses how well a model explains and predicts the future outcomes; also, it is indicative of the level of explained variability in the dataset. The larger the values of R^2, the better fit the model provides, while the best fit exists when R^2 is equal to 1, (b) using the Root Mean Square Error (*RMSE*) that represents the deviation between the predicted and the observed values (the smaller the *RMSE* the more accurate the model is), (c) comparing our models with the baseline (naïve) model, which is the average of the target values (V_{min} or Severity) of the training dataset.

Depending on the target parameter (V_{min} or Severity), we use different *samples* that correspond to information vectors of the values of the dependent and the independent variables that were used to train and test the models. All our training samples come from the characterization phase, in which we gradually reduce the voltage of the chip in 5 mV steps and capture the dynamic events and the system state; in this way, our models take into account the effects of voltage droops. All the independent variables of the samples used to predict the V_{min} consist of the performance counters from the entire execution of the workloads that were normalized per kilo committed architectural instructions, while for the Severity they also include the voltage values of each voltage reduction step. To avoid biasing the results, all the collected values of each performance counter were normalized according to their minimum and maximum values, getting a final value with range [0, 1].

For all the experiments, we split the samples into 80% training and 20% test datasets, while we also cross-validated our models with different combinations of train and test datasets coming from different workloads to detect overfitting (8000K combinations for each experiment). The total population of samples that were used to predict the V_{min} and the Severity is 320 and 800 respectively.

2.3.1. Experimental Results

We implemented our linear regression models with the three different feature selection algorithms targeting both the V_{min} and the Severity in the eight cores of the TTT X-Gene 2 chip running all the workloads from SPEC CPU2006 suite with all their inputs (40 programs in total). In this section, we present only the most representative and interesting cases of our analysis on the most robust (Core 4) and the most sensitive

© 2018. UniServer Consortium Partners. All rights reserved

(Core 0) cores. For all the cases, we evaluate the accuracy when we select from one up to ten most important features to feed our linear model. Note that the addition of more features in the model does not necessarily imply increase in the final accuracy due to the overfitting phenomenon when the model becomes tailored to fit the random noise of the sample rather than reflecting the overall population. Finally, for the population of features with the highest accuracy (the lowest *RMSE*), we present the final prediction model equations according to equation (1).

Predicting V_{min} of the Most Sensitive Core

In Figure **9**, we illustrate the accuracy results (in terms of *RMSE* measured in *mV*) of all our models when we target the V_{min} of the most sensitive core. The black dotted line presents the accuracy of the baseline (naïve) model which is the average values of the training dataset for prediction, while each set of bars represents the accuracy of the different models by using different population of selected features for the prediction. By using more than 4 features, all the methods are less accurate than the baseline model. Moreover, the linear regression model that uses only *RFE* is less accurate compared to the baseline model for all the cases. The linear regression that is accompanied by polynomial transformation with *f_regression* and up to 4 selected features gives better accuracy than the other methods for all the cases. The best accuracy (*RMSE* equal to 5.0108mV) was observed by using the polynomial transformation with *f_regression* selection and only 4 selected polynomial features. Table 3 presents the final prediction model according to equation (1). Moreover, the R^2 that was measured for this prediction model is high (close to 0.75) indicating a good fit of the model.

Our model finally delivers very high accuracy compared to the real values of our experiments (with only 5mV inaccuracy that corresponds to a single voltage reduction step supported by the machine). The potential power savings of using this scenario of prediction and adding a very small guardband of only 5mV (equal to our predicted inaccuracy) are 11.87% compared to the case of using the very pessimistic nominal voltage limit.

In general, the most important features that are selected by the models in Section 2.2.1 can be indicators of large voltage droops (i.e. BTB mispredictions, decode stalls, exceptions, branches) and timing errors in the pipeline of the CPUs (i.e. integer, FP operations). Therefore, they are intuitively well-correlated to the V_{min} values.

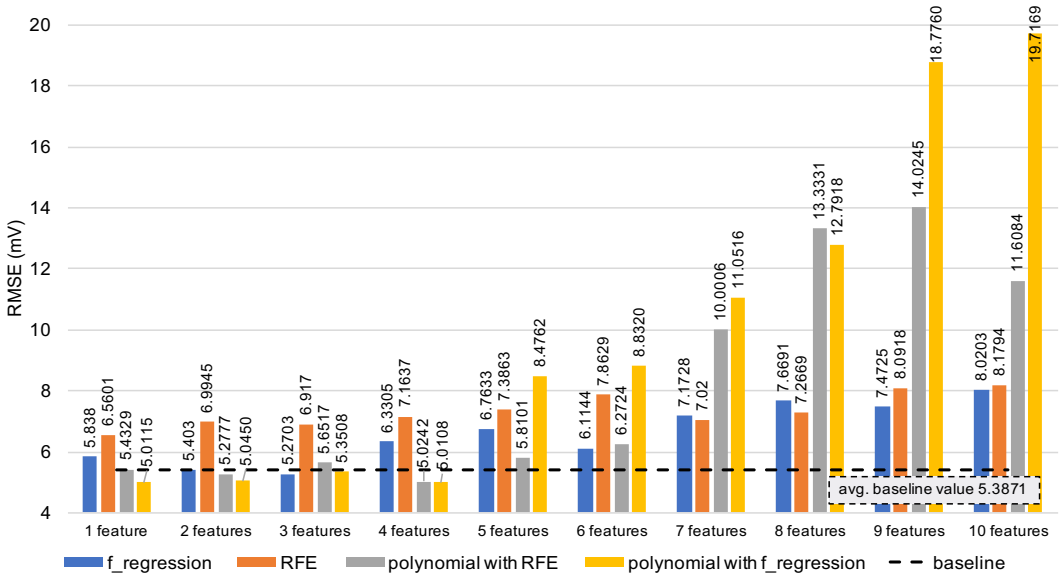

Figure 9: Accuracy of predicting the V_{min} of the most sensitive core.

Table 3: V_{min} Prediction Model of the Most Sensitive Core.

Hardware counters used by the model	Prediction Model (with 4 polynomial features)
L2 data prefetch request (*L2_pref*)	$897.90 + (23.01 * L2\_pref)$

© 2018. UniServer Consortium Partners. All rights reserved

BTB mispredictions (*BTB_miss*)	+ (54.20 ∗ *BTB_miss* ∗ *FP*) – (7.15 ∗ *INT* ∗ *L2_pref*) – (58.84 ∗ *FP* ∗ *Indirect_br*)
Floating point operation (*FP*)	
Integer data processing (*INT*)	
Indirect braches (*Indirect_br*)	

Predicting V_{min} of the Most Robust Core

The results of the accuracy of the different prediction models that target the V_{min} of the most robust core are illustrated in Figure 10. All the models with more than 4 or less than 3 features are less accurate than the baseline model. The best accuracy is observed for linear regression after applying polynomial transformation with *f_regression* and using only 3 polynomial features with *RMSE* equal to 5.0922mV (this model is presented in Table 4). The R^2 is again high (0.70), while the potential power savings of this model are 17.52% compared to the case of using the nominal value.

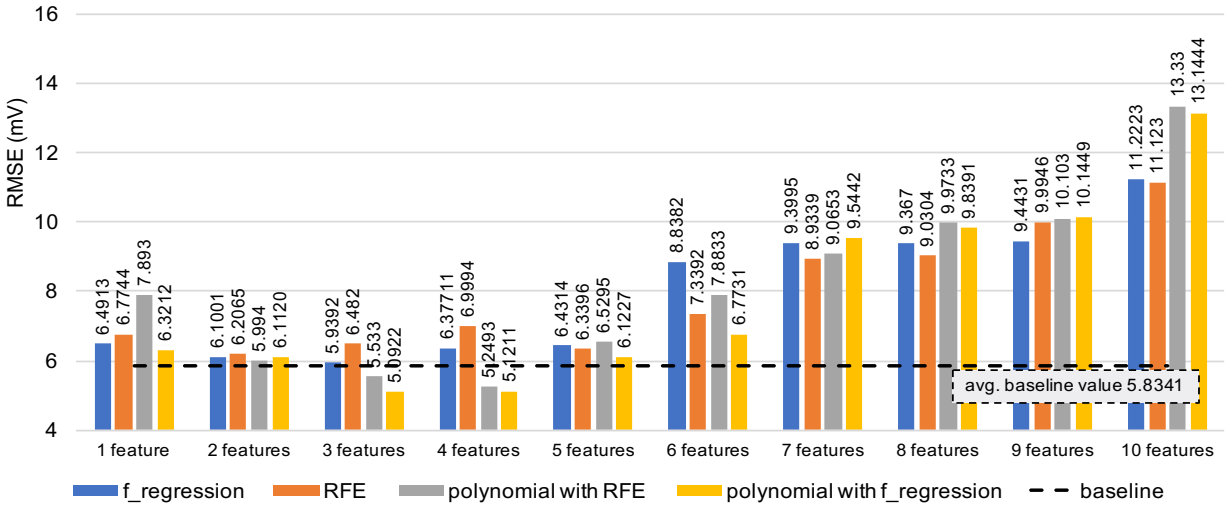

Figure 10: Accuracy of predicting the V_{min} of the most robust core.

Table 4: V_{min} Prediction Model of the Most Robust Core.

Hardware counters used by the model	Prediction Model (with 3 polynomial features)
L2 data prefetch request (*L2_pref*)	898.24 + (27.36 ∗ *L2_pref*) + (61.24 ∗ *BTB_miss* ∗ *FP*) – (8.96 ∗ *INT* ∗ *L2_pref*)
BTB mispredictions (*BTB_miss*)	
Floating point operation (*FP*)	
Integer data processing (*INT*)	

Predicting Severity of the Most Sensitive Core

Figure 11 presents the results of the accuracy (in terms of *RMSE* that is measured in *Severity units*) when we target the Severity of the most sensitive core. Both the linear regression with a simple *f_regression* feature selection mechanism or the model in which we firstly apply a polynomial transformation and then we select the best features with *f_regression* significantly outperform the baseline model for all the cases. The simple *RFE* feature selection and the model with the polynomial transformation with *RFE* feature selection are less accurate than the baseline model for all our experiments. The best accuracy (*RMSE* equal to 2.7223 Severity units) is observed for the model that uses *f_regression* feature selection with 3 features (this model is presented in Table 5). The R^2 for this model is equal to 0.92 indicating model's efficiency. The potential power savings of using this aggressive prediction scheme that targets the Severity including the measured error are 16.59% compared to the case of using the nominal voltage value for protection. These savings correspond to 39.76% more power savings instead of using the conservative prediction targeting the V_{min}.

© 2018. UniServer Consortium Partners. All rights reserved

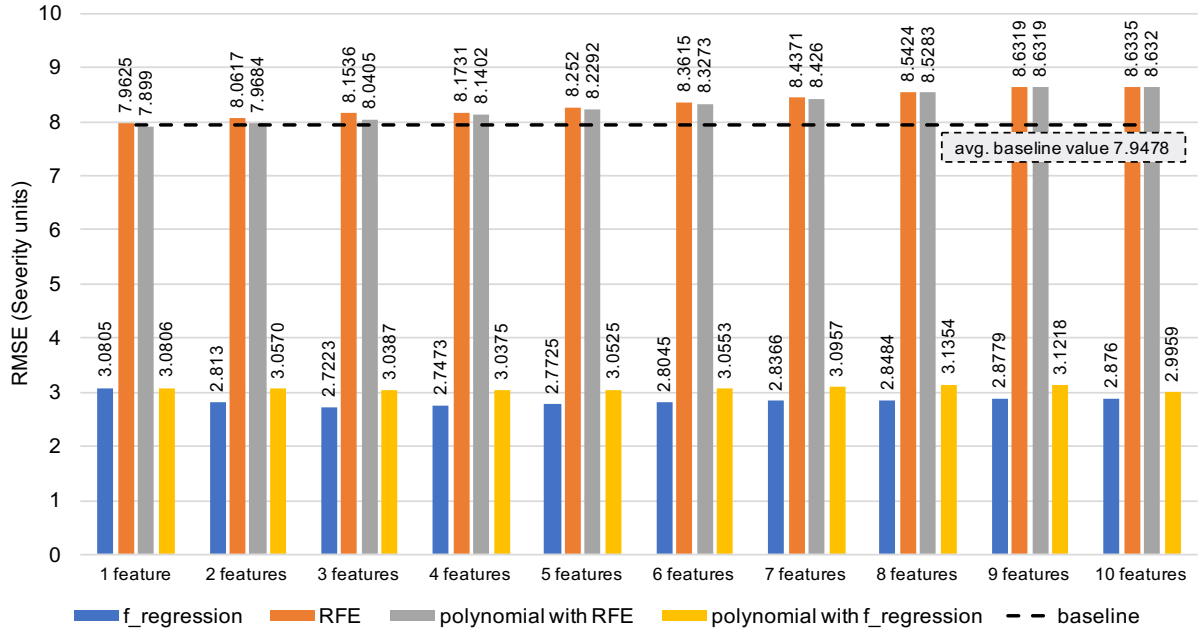

Figure 11: Accuracy of predicting the Severity of the most sensitive core.

Table 5: Severity Prediction Model of the Most Sensitive Core.

Hardware counters used by the model	Prediction Model (with 3 features)
Voltage (*Volt*)	20.35 + (3.66 ∗ *Volt*) – (2.34 ∗ *L1D_tlb_write*) – (22.53 ∗ *dec_stalls*)
L1 data TLB write (*L1D_tlb_write*)	
Decode stalls (*dec_stalls*)	

Predicting Severity of the Most Robust Core

Finally, Figure 12 illustrates the results of the accuracy of the models that predict the Severity values of the most robust core of the chip. All the models apart from the case of polynomial transformation with *RFE* feature selection outperform again the baseline prediction (the difference is about 5.2 Severity units). The best accuracy in terms of *RMSE* is equal to 2.5 Severity units for the linear regression model with *RFE* feature selection using 6 features (this model is presented in Table 6). The R^2 for this model is again very high (equal to 0.91). Including the error, the potential power savings are 20.28% instead of using the nominal voltage value, while these gains correspond to 15.75% more power savings compared to the conservative case of using a scheme that targets the V_{min}.

© 2018. UniServer Consortium Partners. All rights reserved

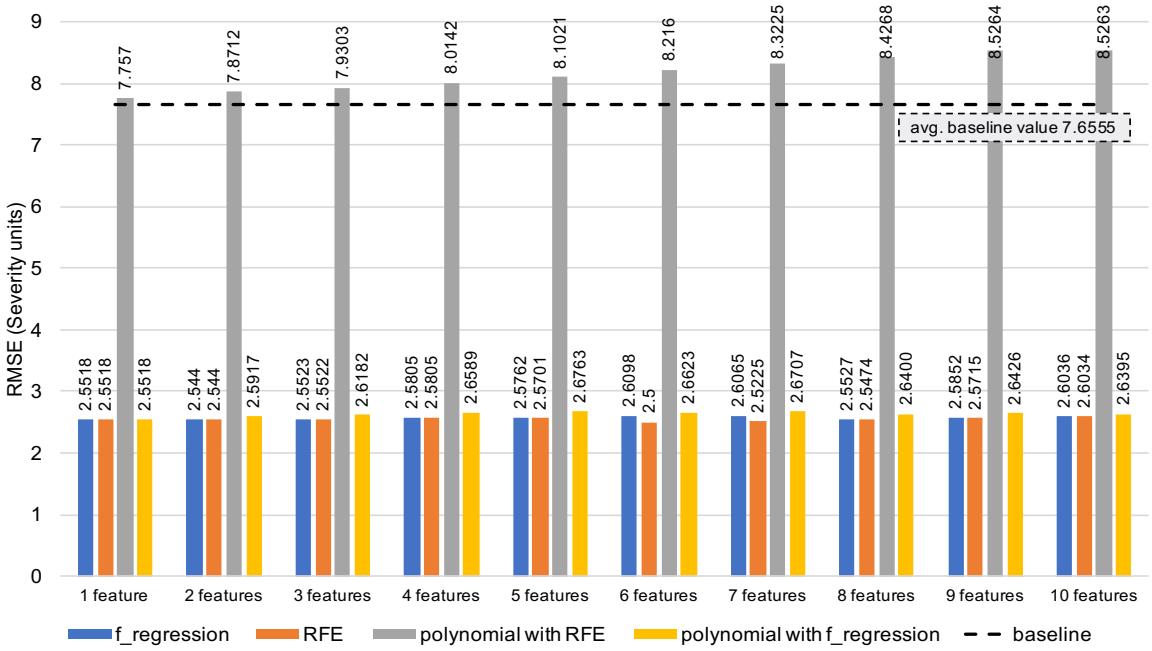

Figure 12: Accuracy of predicting the Severity of the most robust core.

Table 6: Severity Prediction Model of the Most Robust Core.

Hardware counters used by the model	Prediction Model (with 6 features)
Voltage (*Volt*)	18.94 + (47.80 ∗ *Volt*) – (20.56 ∗ *excpt_taken*) + (7.09 ∗ *L1I_miss*) – (3.92 ∗ *cond_br*) – (29.50 ∗ *L1D_tlb_write*) – (22.11 ∗ *dec_stalls*)
Exceptions taken (*excpt_taken*)	
L1 instruction cache miss (*L1I_miss*)	
Conditional branches (*cond_br*)	
L1 data TLB write (*L1D_tlb_write*)	
Decode stalls (*dec_stalls*)	

© 2018. UniServer Consortium Partners. All rights reserved

3. Characterization of Dynamic Memories

Apart from the cores and the on-chip caches, Dynamic Random-Access Memory (DRAM) is an important subsystem of computing systems that has recently attracted a great amount of interest as a target for power and performance optimizations. The density of DRAM devices which has been increasing steadily the past few decades, coupled with our increased needs for main memory capacity makes DRAM a significant contributor in the total power budget of contemporary computing systems [7]. Moreover, further scaling of DRAM cells is accompanied by reduction of cell reliability and increased cell leakage leading to high refresh rates, which in turn inflates power consumption [8] [9] [10] [11] [12]. As a result, any power optimizations regarding DRAM devices has a significant impact [13].

DRAM power could be reduced through lowering the supplied voltage or relaxing DRAM refresh rate, however this could compromise consistency of data stored in DRAM. This rate determines how often each cell is refreshed, which is required to sustain data stored in DRAM due to its dynamic nature. Refresh operation consumes power that is expected to be up to 50% of the total power consumption of the whole DRAM power in next generations of density [14].

In this deliverable, we present the results of the experiments with DRAM operating at the lowered input voltage and relaxed refresh rates. We discovered the major parameters affecting DRAM faults. In order to extend the experiments, we have implemented a thermal controlled environment to test the memory under a range of temperatures and realized a heterogeneous reliability framework on X-Gene 2.

3.1. Experimental Setup

Our experimental setup is based on the two X-Gene 2 validation boards that have been delivered to QUB by APM. In D4.1, APM provided a brief description of the X-Gene 2 firmware and specification of i2c interfaces to control the main operational parameters of various components inside the development boards, such as supplied chip and DRAM voltages, refresh rate of DRAM and access to sensors. Furthermore, Linux kernel of X-Gene 2 can report all DRAM and cache errors registered by ECC in *syslog*. The kernel receives information about ECC errors from the firmware, and all errors are registered asynchronously. The kernel reports information about source of an error, including DRAM, cache, MCU, bank, rank and so on. It also provides information about the type of error: correctable (single-bit error which could be corrected by ECC) and uncorrectable (double-bit errors which could be only registered by ECC).

3.1.1. Framework for thermal stressing

Temperature plays a major role in the reliability of the DRAMs as shown by previous research works [15] [16] and validated by our experiments, so to avoid any effects of the variable temperature of the DIMMs, we need to control the temperature of the server. Previous works have achieved this either by setting the machine in a thermal oven [15] or having an element on each DIMM controlling the temperature [17].

Our approach is based on the second scheme, in order to control the temperature on the DIMMs, we have developed a custom adapter that fits on top of each side of the DIMM. The control board (Figure 13) is 1U rack-mountable that has a power supply, a Raspberry Pi 3, for network control of the system, and four Carel ir33, two-channel temperature PID controllers that have NTC sensors and two solid state relays to control the current on a resistive element that is mounted on the both sides of the DIMM. The adapter consists of a resistive element and thermally conductive tape transferring the heat of the element to all the chips of the DIMMs in uniform way. Figure 14 shows the X-Gene 2 board fitted with four DIMMs with the custom adapter and the sensors.

We measure the temperature of the chips through the sensor embedded on the SPD and directly on the heating elements. The controller of the elements is a PID controller, and we have observed a deviation from the set temperature of +/-0.8 °C.

Figure 15 shows the X-Gene 2 server from above with a thermal photograph when on the board, the thermal adapters do not operate. In this way, we can identify the hotspots around the board and observe the difference of temperature of each DIMM.

© 2018. UniServer Consortium Partners. All rights reserved

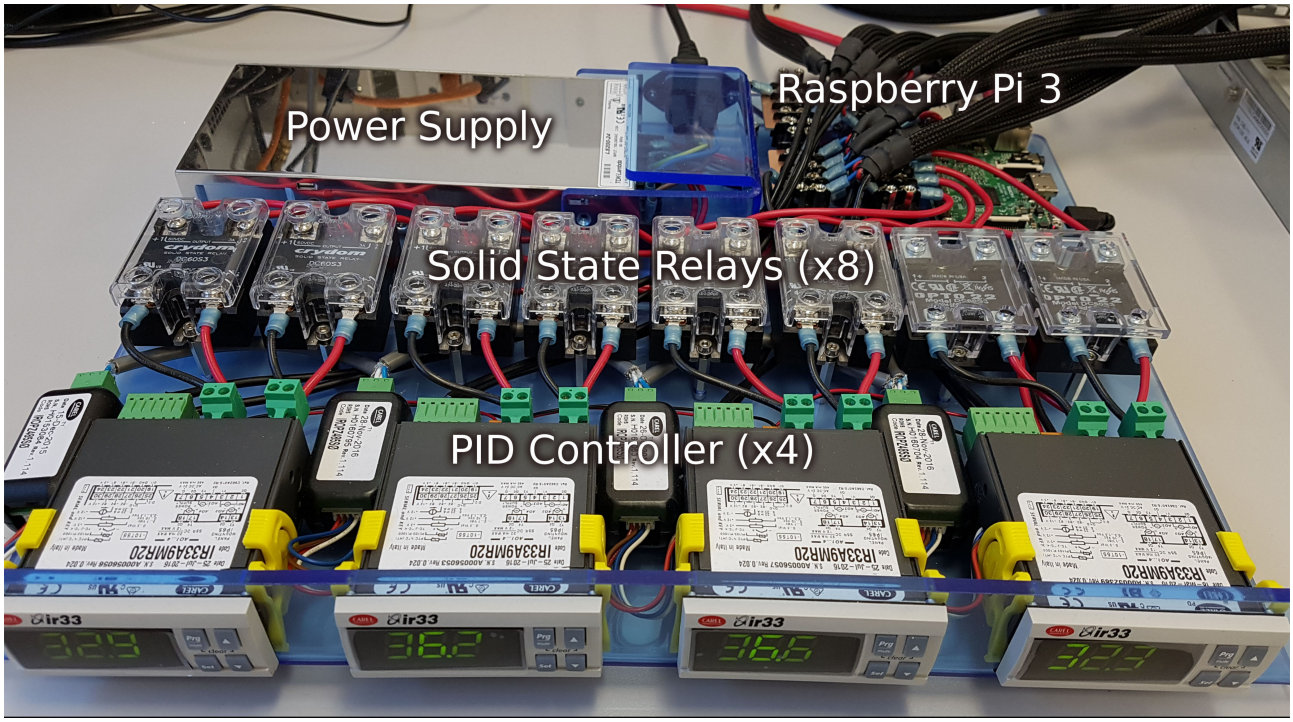

Figure 13: Control board of the thermal framework.

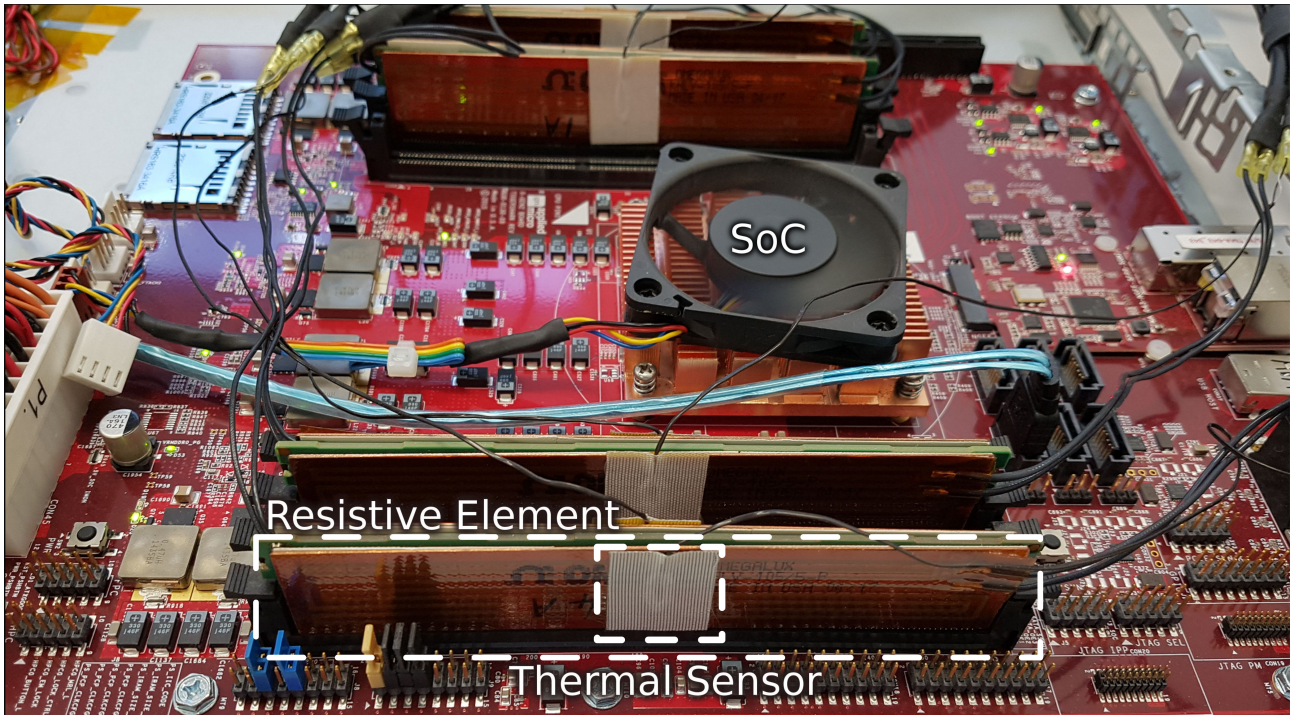

Figure 14: X-Gene 2 with 4 DIMMs with thermal adapters, consisting of a resistive element and a thermal sensor.

© 2018. UniServer Consortium Partners. All rights reserved

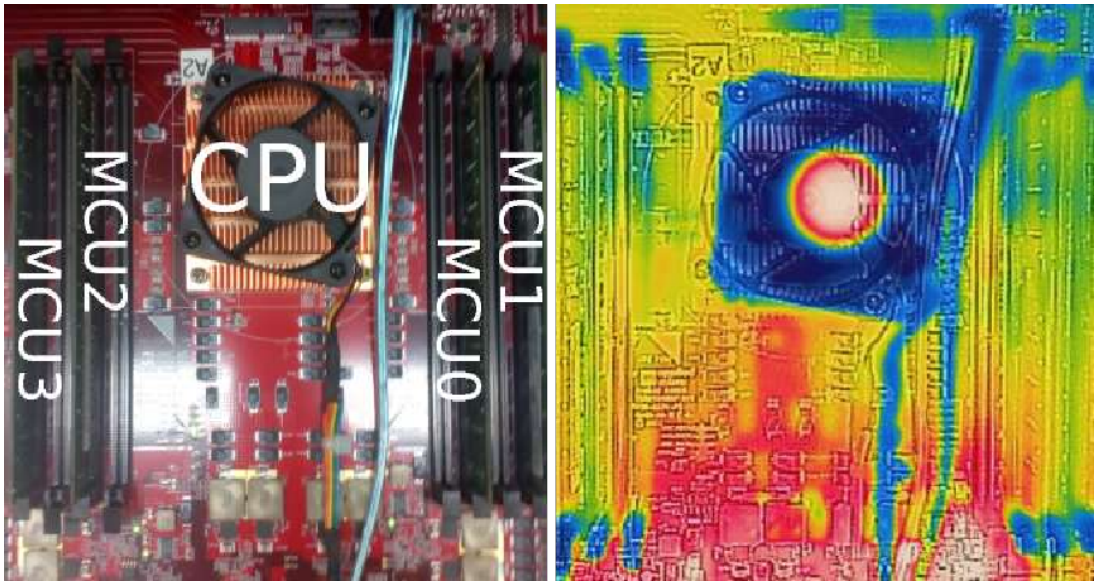

Figure 15: Photograph of X-Gene 2 with annotation of components and equivalent thermal photograph pinpointing the difference in temperature of DIMMs

3.1.2. Framework controlling allocations

As previously mentioned on the D3.3, we had explored relaxation of the refresh rate on a dual-socket Intel server, in order to try an extended range of refresh rate. With that setup, we had the capability to split the memory into two domains, one reliable with the nominal settings applied, while on the second one, we can control the parameters of the memory and have variable reliability. This was possible because there were two separate memory controllers.

The X-Gene 2 has two Memory Controller Bridges (MCBs) that each one controls two Memory Controller Units (MCUs), for a total of four MCUs. Voltage can be controlled on the granularity of the MCBs, while DRAM parameters, such as refresh rate, CAS latency, row precharge and active time, can be configured independently on each MCU. This gives to the X-Gene 2, the ability to control on fine granularity and on demand for each memory channel the parameters.

A limitation of the applicability of this type of control was the interleaving of memory across the MCBs and MCUs. Interleaving is commonly used in memory controllers to utilize the maximum out of the total bandwidth of the channels by spreading consecutive addresses and accesses across the memory channels. This can cause issues, if you are setting different refresh rate in the MCUs, then the state of each MCU could be different, for example one of the MCUs could refresh a line while the others are idle. APM provided support for the firmware of the X-Gene 2 and enabled in the firmware the option to select the level of interleaving in the memory system, having the option for enabled interleaving between MCBs and MCUs, enabled interleaving only inside each MCB or disabling interleaving completely. This allow us to exploit the fine-granularity of setting the parameters of each MCU independently.

In order to accommodate the changes of the interleaving and in consequence of the hardware in the software layer, changes in the Linux kernel are needed. During this period APM also provided us with the latest Linux kernel supporting the X-Gene, specifically the version 4.11. We have made modifications in this version of the kernel and we can make zones of memory at the boot-up, for each MCU. We can then restrict the kernel and the applications from using by the default memory from one of the zones, that MCU then could be configured with settings that affect the reliability without affecting the critical parts of the system. Figure 16 shows an example of the memory address of the two MCBs that is not interleaved, that can be used for different parts of the application. Specifically, it is shown that the MCB 1 is configured with relaxed parameters, and considered variably-reliable as errors could appear in this domain.

© 2018. UniServer Consortium Partners. All rights reserved

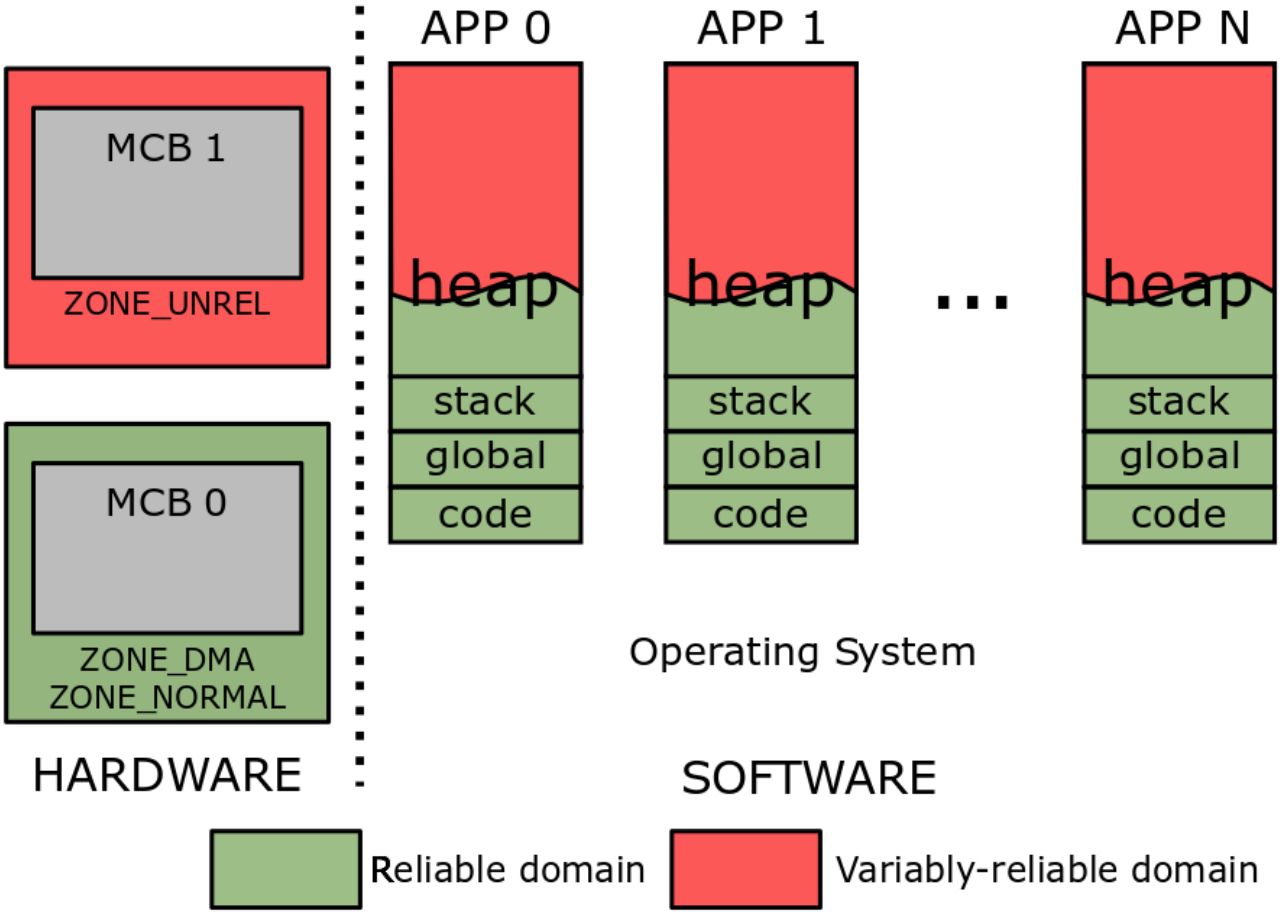

Figure 16: System setup for reliability domains realized in the X-Gene 2.

3.2. Experiments

For different experiments, we modify the refresh rate and the voltage independently, or as well as simultaneously. The X-Gene 2 platform allows us to control refresh rate for each MCU, up to a maximum refresh period of 2.283s, while the nominal one is 64ms. The platform also allows to configure voltage of each MCB separately. The MCU0/1 (MCB0) and MCU2/3 (MCB1) could be operated at different voltage levels. The nominal supply voltage for each MCB is 1.5V, while we found the minimum supply voltage to be 1.428V, as lower values of voltage make the system instantaneously non-functional.

We have experimented with a set of benchmarks, ranging from micro-benchmarks that are used typically in DRAM characterization to real-world applications. Each one of those stresses the cores and DRAMs in a different way and allows to study the influence of application specific characteristics i.e. access patterns and used number of threads on DRAM reliability either directly or indirectly, by studying their impact on the incurred SoC activity and DRAM temperature. In particular, we chose: Data pattern micro-benchmarks which are designed to stress the retention time while exploiting the impact of neighbouring cells. As previous researchers suggested, we have used the following patterns: all 0s, all 1s, checkerboard and random. Rodinia Benchmark Suite represents a variety of HPC algorithms that are used for benchmarking parallel computers. We chose to use backprop, nw, srad and kmeans to cover a range of domains, i.e. Machine Learning, Bioinformatics, Image Processing and Data Mining. To evaluate how parallelism and processing power affects the characterization, we have executed these benchmarks with 1 and 8 threads. As a real workload, we are using the WSE DoSSensing, operating with 4 threads. And finally, CloudSuite [18] is suitable workload for the X-Gene2 as ARM based servers target to substitute Intel servers in Cloud and datacenter. To cover popular server workloads, we have ported three benchmarks from the CloudSuite to ARMv8 which are being deployed through containers in Docker: Memcached [19], Graph Analytics [20] and Web Search [21].

© 2018. UniServer Consortium Partners. All rights reserved

3.2.1. Results on nominal conditions of temperature

The initial purpose of our experiments is understanding if memory reliability operated under relaxed refresh rate could be characterized with a set of micro-benchmarks. To follow this, we should identify all cells that have small retention time, and thus are more prone to failures. We do this by executing data pattern micro-benchmarks ensuring that we cover all memory available for the user space.

Figure 17**Error! Reference source not found.** shows the spatial and density distribution of errors between different memory locations. Identification of memory locations where ECC discovered errors are provided on the left, while the distribution of locations between MCUs is shown on the right. The darkness of the color indicates the number of errors registered per 8 hours for a certain memory location, while the color identifies the experiment, blue reflects an experiment with relaxed refresh rate, red for an experiment with simultaneous relaxation of refresh and voltage, and if the errors were detected in both of the experiments, we mark them with green color.

We observe that the random micro-benchmark identifies more locations than the other micro-benchmarks, which is in agreement with the observations made also by Liu [15]. In addition, in all our experiments with the server operated under the nominal conditions, we observe only single bit errors, which were corrected by the available SECDED ECC. Based on such an analysis, we reach similar conclusions as existing studies [15], refresh rate can be aggressively relaxed, in our case up to 43x, from 64 ms to 2.283 s. For those experiments, the probability of erroneous 64-bit word is around 10^{-9}, which can be handled by ECC, as long as two bit errors do not happen on the same word.

To investigate if the above conclusion is accurate enough and the micro-benchmarks are able to reveal all weak cells, we executed the four memory-intensive Rodinia benchmarks under relaxed parameters. In these experiments. We execute a single threaded version and a parallel version with 8 threads to evaluate how parallel access patterns affect the distribution of errors for each benchmark. In Figure 17**Error! Reference source not found.**, the results of the experimental campaign are evident that the single threaded version of each benchmark triggers similar number of memory errors at the same locations as the ones observed when executing the micro-benchmarks.

In addition, we observe that the parallel Rodinia benchmarks are able to stress and reveal more memory locations triggering errors that have not been discovered by the micro-benchmarks and the single threaded versions. This leads to a significant observation that the parallel versions of the benchmarks do affect the system and the reliability, while significantly changing the spatial distribution of errors. Each benchmark has some unique locations that are not activated by the other benchmarks. These results complement the ones extracted from Rodinia, leading to the following observation. At the normal operating conditions, data pattern micro-benchmarks are not as effective in detecting weak cells as real workloads when DRAM is operated with relaxed refresh rate.

We experimentally identified the lowest operating DRAM supply voltage at 1.428 V after which most probably the circuitry of DRAM stops working. This voltage level is higher than the minimum supply voltage of DIMM specification [22]. In Figure 17**Error! Reference source not found.**, we can observe how the spatial and density distribution of errors differs for the experiment under relaxed refresh rate and voltage. When we also lower the voltage, ECC reports errors at locations that have not been discovered before. We clearly see that if we lower the supply voltage while having relaxed refresh rate, it significantly increases dispersion of error locations. We suggest that this experiment manifests more errors due to cells which fail being less charged under lowered voltage.

© 2018. UniServer Consortium Partners. All rights reserved

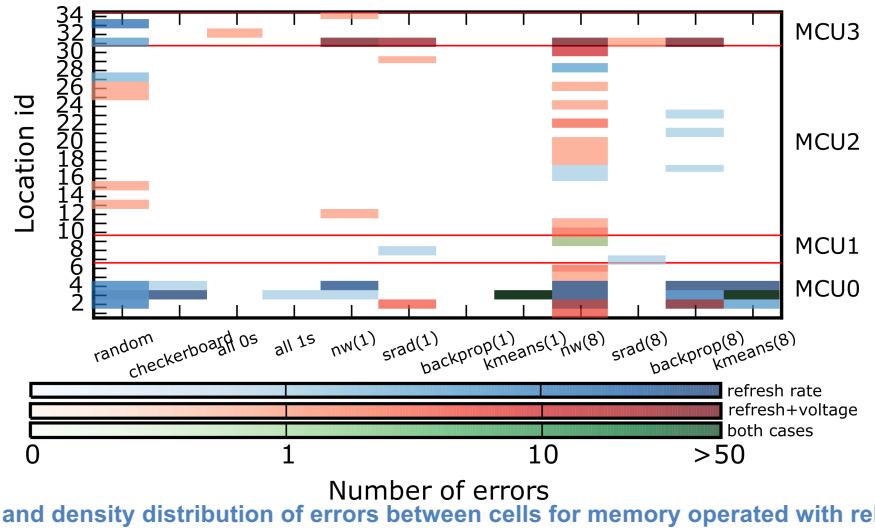

Figure 17: Spatial and density distribution of errors between cells for memory operated with relaxed refresh only (blue), relaxed refresh and lowered supply voltage (red) and if the error occurred in both scenarios (green).

Figure 18 shows the same spatial and density distribution of errors for memory operated with relaxed refresh rate and lowered supply voltage comparing the CloudSuite benchmarks and the micro-benchmarks. It is obvious that the number of errors located while executing the benchmarks are much higher than previously discovered by the micro-benchmarks or the Rodinia. As it is not yet clear why the different benchmarks are able to identify more locations than the micro-benchmarks, we continued our research trying to identify the parameters that are affecting the reliability of the whole system when operating under a real hardware platform and realistic workloads.

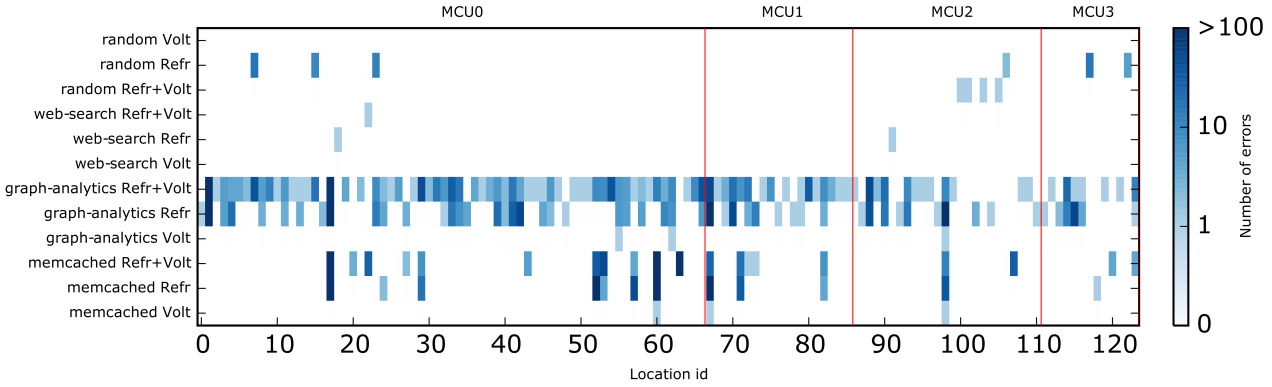

Figure 18: Spatial and density distribution of errors between cells for memory operated with relaxed refresh and lowered supply voltage.

One significant finding is that the WSE DoSSensing application do not excite the memory subsystem and we do not observe any correctable error during the experiments relaxing the refresh rate or both relaxing the refresh rate and the supply voltage. This could be explained based on the data from the next Section, where we try to identify the application parameters that are affecting the reliability of DRAM.

3.2.2. Performance indicators and memory access patterns

The difference in the distribution of errors, that was found previously in D3.3 and in the previous Section, may be related to benchmark characteristics. To explore the impact of the characteristics among benchmarks, such as data and frequency memory access patterns, and other performance characteristics, such as IPC and the CPU utilization, on the distribution of errors, we collect statistics for the performance indicators denoted in using *perf* tool. Particularly, we investigate the correlation between the indicators, the total number of memory errors and the number of unique error locations reported for our experiments with refresh rate and voltage using the Spearman's rank correlation coefficient which characterizes the monotonic relationship between two variables. Table 7 shows the correlation coefficient *r*, which denotes the strength and direction of the correlation, and *p*-value for a hypothesis test whose null hypothesis (H_0) is that two sets of data are not correlated, i.e. the probability that the variables are not correlated.

© 2018. UniServer Consortium Partners. All rights reserved

There is no evidence of the correlation between the performance indicators reflecting memory access patterns and the number of errors or the number of unique error locations. However, we found that this correlation is very strong for the CPU utilization indicator: *ρ*-value for the total number of errors is *0.0003* and *0.052* for the number of unique error locations (see Table 7). Nonetheless, it is not clear why the correlation is observed only for the CPU utilization but not for other performance parameters.

Table 7: Spearman's rank correlation coefficient and ρ-value for various performance indicators.

Event	Relaxed Refresh Rate					Lowered Voltage		
Pattern	Number of errors			Number of unique error locations		Number of errors	Number of unique error locations	
	r	*P*	*r*	*ρ*	*r*	*ρ*	*r*	*ρ*
IPC	0.333	0.318	0.257	0.445	0.501	0.116	0.065	0.849
L1 read	0.287	0.392	0.166	0.627	0.469	0.145	0.014	0.967
L1 write	0.187	0.582	0.078	0.819	0.219	0.518	-0.182	0.592
L1 accesses	0.305	0.361	0.221	0.514	0.46	0.154	-0.019	0.957
L2 read	0.187	0.582	0.156	0.646	-0.137	0.689	0.154	0.651
L2 write	0.196	0.564	0.087	0.798	0.21	0.536	-0.173	0.612
L2 accesses	0.333	0.318	0.257	0.445	0.501	0.116	0.065	0.849
Mem read	0.068	0.842	0.17	0.617	-0.21	0.536	0.252	0.455
Mem write	-0.159	0.64	-0.097	0.778	-0.383	0.245	0.089	0.796
Mem accesses	-0.064	0.852	0.041	0.904	-0.301	0.369	0.182	0.592
CPU utilization	0.69	0.019	0.882	0.0003	0.506	0.112	0.598	0.052

On top of the hardware counters of the application, it is already known that the temperature of DIMMs affects DRAM reliability and thus the spatial and density distribution of errors. To investigate this, we have measured the temperature by using on-board sensors provided for each DIMM, while having the server on a typical server rack with forced ambient temperature of 18°C without the thermal adapters.

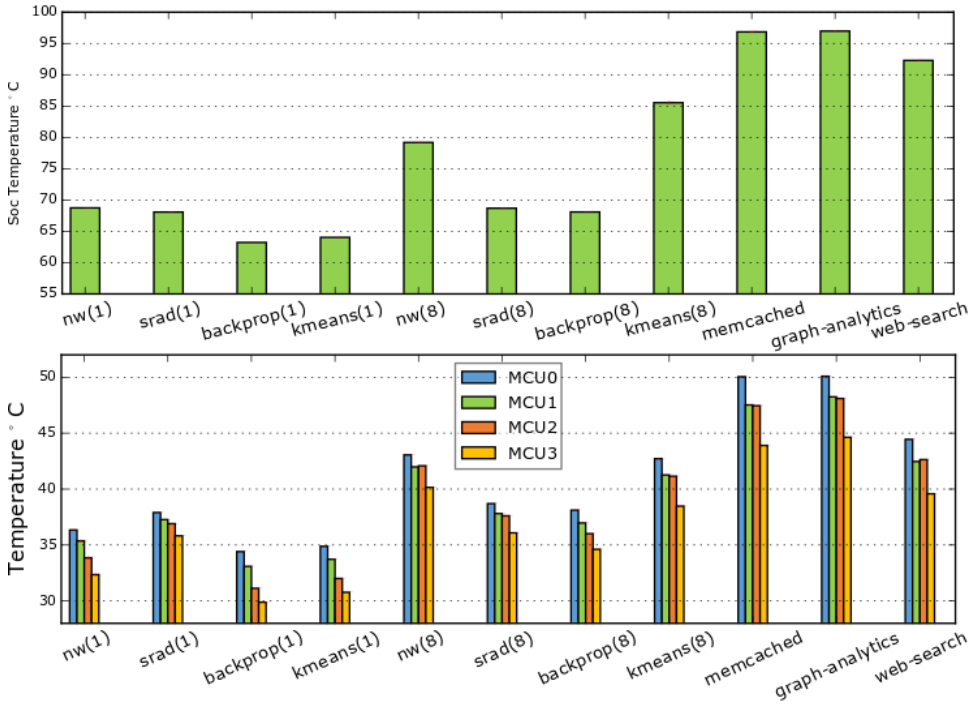

Figure 19: Average temperature of the core and each DIMM across the duration of execution of the benchmark.

© 2018. UniServer Consortium Partners. All rights reserved

The top part of Figure **19Error! Reference source not found.** shows the average temperature of the SoC for the period of the 8-hour experiments with 95% confidence intervals and the bottom the corresponding average temperatures for each DIMM. The highest temperature averaged over all DIMMs was measured for the graph-analytics benchmark which also incurred the highest dispersion of error locations in memory during our experiments under relaxed refresh rate and voltage as explained in the previous Section. Further analysis of the correlation between the temperature and the number of errors can be seen on Table 8, presenting the Spearman's rank correlation.

Table 8: Spearman's rank correlation coefficient and ρ-value for DRAM and SoC temperatures.

Event	Relaxed Refresh Rate				Lowered Voltage			
	Number of errors		Number of unique error locations		Number of errors		Number of unique error locations	
	r	*ρ*	*r*	*ρ*	*r*	*ρ*	*r*	*ρ*
SoC temperature	0.232	0.492	0.437	0.1792	0.592	0.055	0.714	0.0137
DRAM temperature	0.601	0.050	0.763	0.0063	0.378	0.252	0.825	0.018

Note also that in our experiments, MCU0 was found to have the highest temperature, which explains the fact that the majority of errors took place in MCU0. Figure 20 shows the spatial and density distribution of the errors between MCUs and ranks when we relax the refresh rate and lower the voltage in the case of Rodinia benchmarks, single- multi- threaded and the CloudSuite benchmarks. In Figure 20 we present the distribution as a polar plot where *θ-axis* specifies MCUs and ranks, while *ρ-axis* reflects the number of errors.

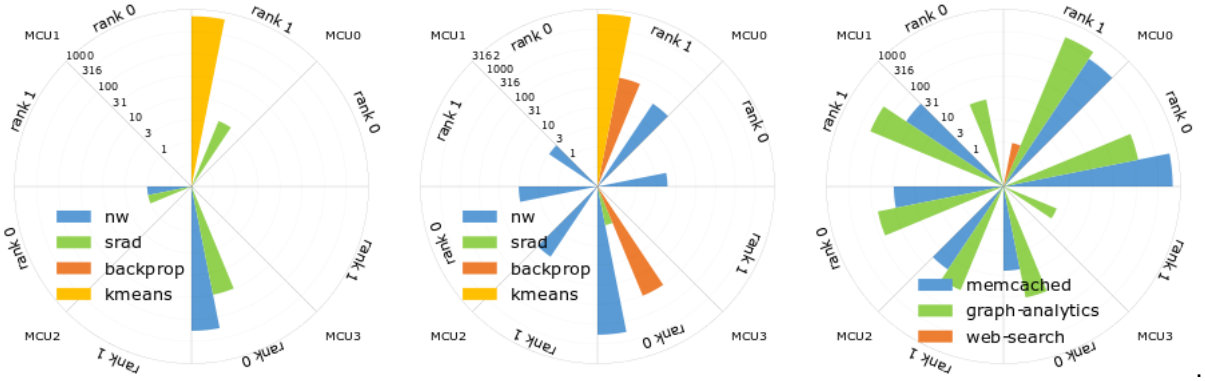

Figure 20: Distribution of the number of errors across each DIMM and rank.
i) Rodinia single-thre aded, ii) Rodinia multi-threaded, iii) CloudSuite.

3.2.3. Results with thermal stressing framework

As it is obvious from the previous experiments that the reliability of the machine is heavily affected by the temperature, we continue on with experiments on the thermal framework. The DRAM temperature across different benchmarks vary due to the correlation with the workload characteristics and SoC temperature as shown on the previous section. It is important to test the DIMMs under a range of temperatures, simulating different ambient temperatures as the characterization needs to be able to identify critical conditions. Micro-server at the Edge of Cloud should be able to sustain operation under non-ideal conditions so it is needed to characterize their reliability on the extreme margins.

Furthermore, working with fixed temperatures of the DIMMs will help in identifying the data and frequency patterns of the benchmark that contribute in having more errors while avoiding any indirect effect of the benchmarks, such as the CPU utilization, on the temperature of the DIMMs. We are using the thermal framework described in Section 3.1.1 for characterization of the X-Gene 2 at fixed temperatures for the DIMMs.

© 2018. UniServer Consortium Partners. All rights reserved

We are starting the characterization with the WSE DoSSensing application, we are executing it for 400 executions with four parallel instances, under controlled temperature of 40, 50, 60 and 70 °C and we observe the correctable errors reported by the ECC as shown in the Table 9. Furthermore, when we analyse the number of unique weak cells across the duration of the experiment, distinguishing errors in 400 runs (Table 9**Error! Reference source not found.**), we can clearly see that the majority of the errors occur in the first part of the application.

Generalizing this based on extended experiments with multiday executions, we observe that in the first 25% of the application's execution duration, we discover more than 80% of the total weak cells. As it will be described in more detail in one of the next Sections.

Table 9: Testing WSE DoSSensing under variable temperature.

Temperature (°C)	Total Number of Errors	Number of Unique Errors
40	4	4
50	1119	13
60	36793	653
70	313953	7010

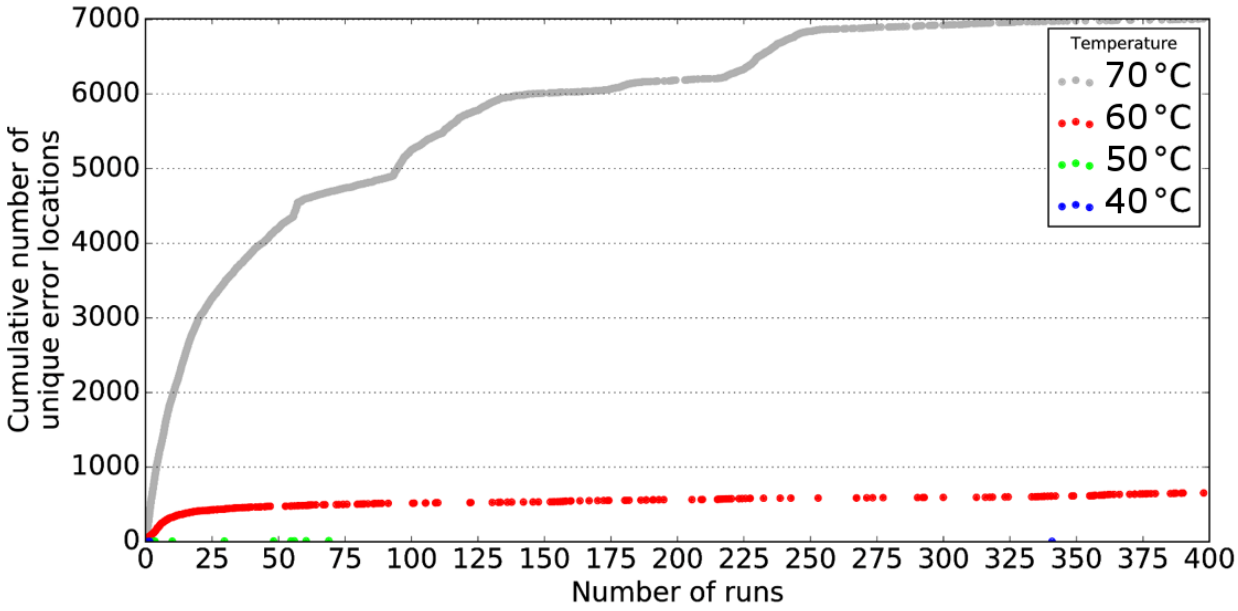

Figure 21: Unique weak cells across the duration of the experiment.

Table 10 shows the distribution of the unique weak cell locations that were discovered and corrected by ECC during the experiments of the WSE DoSSensing benchmark at 70°C. For the reported errors, we have experiments with combined both relaxed refresh rate and lowered supply voltage.

Table 10: Distribution of errors in different device levels.

Event	Number of Unique Weak Cell Location per device (DIMM/rank/bank)								Max difference (%)
Across DIMMs	1626		2385		1453		1546		39.07 %
Across Ranks	3411				3599				5.22 %
Across Banks	796	922	884	957	852	839	817	943	15.58 %

© 2018. UniServer Consortium Partners. All rights reserved

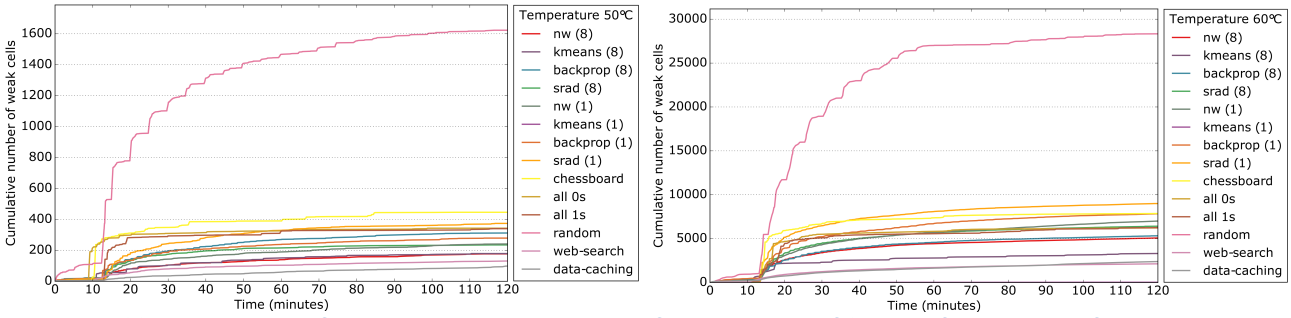

Figure 22: Distribution of errors across the duration of experiments of 2 hours for relaxed refresh rate and voltage under an enforced DRAM temperature of 50°C and 60°C.

Moving on with the micro-benchmarks, Rodinia benchmarks and the CloudSuite, Figure 22 shows the results of the discovered correctable ECC errors for experiments of two hours. The experiments were conducted in shorter duration based on the findings of a next Section and because the restrictive nature that it would time months of machine time to execute the full experiments in many different temperatures and parameter settings of refresh and voltage. Form the data, we observe that it is clear that the random micro-benchmark is the best at discovering the weak cells location as was previously reported in the [15] when the conditions of temperature are controlled and not affected by the CPU utilization of the benchmark.

One other observation that could made is that for different temperatures the order that benchmarks do rank based on the number of weak cells discovered is not the same which could be affected by other parameters.

It is interesting to see that across our experiments with real-workloads, we show that an increase of 10°C on the DIMMs leads to an order of magnitude increase in the total number of errors and the number of unique weak cells discovered. This is according to the findings of other work [15] that their experiments are based on microbenchmarks alone.

Similarly to the nominal conditions of temperatures, lowering only the voltage of the DRAM do not produce much errors, in the order of tens, and could be considered negligible compared to the hundreds of unique locations at 50°C and thousands at 60°C, however in the experiments where the lowered supply voltage is combined with relaxed refresh the behaviour of the manifested errors gets affected.

3.2.4. Optimization of the experiment duration

One of the targets in our research is to investigate how much time is required to characterize reliability of DRAM operated under relaxed refresh rate and lowered supply voltage for a specific benchmark. To achieve a good coverage of weak cells [16] and high confidence for our results, we run each benchmark for 8 hours in our initial characterization. We choose this period of testing to make our results comparable with [16] where authors use 1000 rounds in their micro-benchmarks which correspond to 8 hours of running of these micro-benchmarks on the APM platform. We run Rodinia and CloudSuite benchmarks in a loop until the total execution time for each benchmark is 8 hours.

To reduce the total time of the experiments, we analyse the distribution of the discovered error locations in time to find the optimal trade-off between the number of discovered distinct error locations and running time for each benchmark. Figure 23-a demonstrates how the total number of distinct error locations discovered using all running benchmarks changes with time of refresh rate experiments. Particularly, the left y-axis shows the percentage of distinct error locations discovered from the beginning of experiments up to a certain moment, *t*, accumulated over all benchmarks, i.e.

$$Coverage(t) = \frac{\sum_0^t Number_{locations}}{\sum_0^{8hours} Number_{locations}}$$

While the right y-axis shows the percentage of distinct error locations discovered for a certain period of time *[t, t+10minutes]* accumulated over all benchmarks, i.e.

© 2018. UniServer Consortium Partners. All rights reserved

$$Percentage_{locations}(t) = \frac{\sum_{0}^{t+10minutes} Number_{locations}}{\sum_{0}^{8hours} Number_{locations}}$$

Figure 23**Error! Reference source not found.**-b depicts the same statistics for our experiments with relaxed refresh and voltage. We see that benchmarks cover about 80% (84.27% for experiments with relaxed refresh and 76.26% for experiments with relaxed both) of error locations for 120 minutes, which is 25% of the total time of one application run. Moreover, in both figures we see that $Percentage_{locations}(t)$ does not exceed 3% after running benchmarks for 120 minutes. Following these results, we reduce the total time of all experiments to 2 hours and presumably cover about 80% of all error locations which may be discovered 8 hours of these experiments.

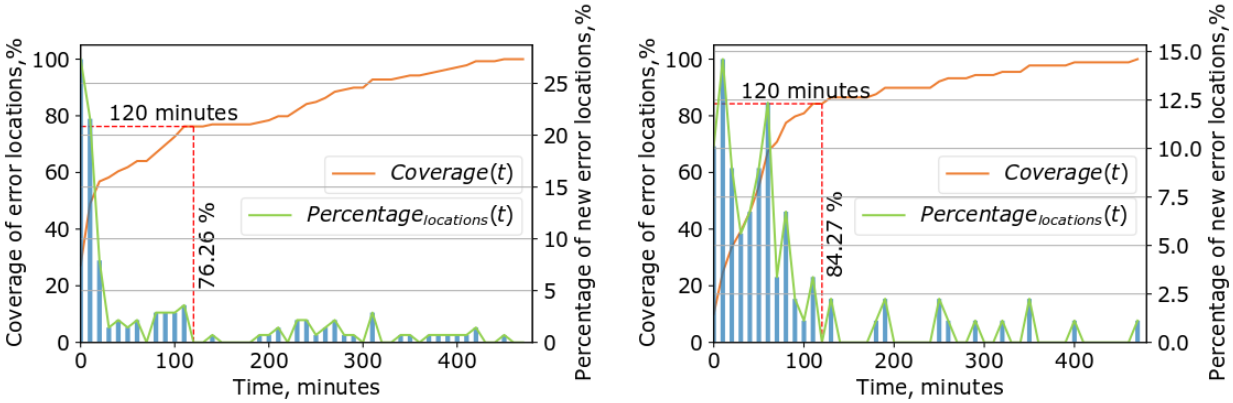

Figure 23: Coverage of microbenchmark across the duration of the experiments for a) refresh rate and voltage, and b) refresh rate only.

3.2.5. Effectiveness of benchmarks

We can compare the effectiveness in discovering the weak cells of each benchmarks based on the number of locations of weak cells discovered. We accumulate the discovered locations of weak cells from the experiments in a specific temperature of all the application and for each application we calculate the percentage of the total that are discovered by the application. The results about the random micro-benchmark in Figure 24 show that the coverage compared to the total number of locations for all the benchmarks is considerably high at more than 80% of the total locations. So, it is adequate to cover the majority of the locations without having to run the experiments with the rest of the benchmarks.

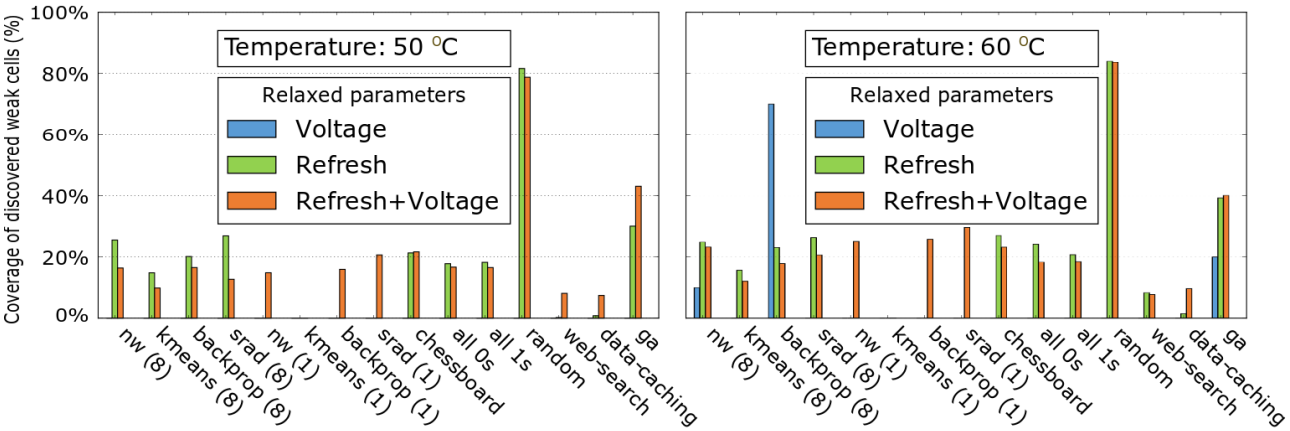

Figure 24: Coverage of the total locations of the errors discovered by each application at two temperatures.

3.2.6. Effectiveness of existing Error Correcting Codes (ECC)

During our experiments, ECC is activated for detecting and correcting of the errors, allowing us to evaluate also its efficacy as an error mitigation mechanism. SLIMpro monitors the SoC and reports ECC errors

© 2018. UniServer Consortium Partners. All rights reserved

detected by hardware to the Linux kernel through delivering an asynchronous message via an i2c mailbox interrupt. The SLIMpro reports to the kernel: the source of an error at the level of the MCU, rank/bank, row and column which point to a word in memory where the error has been detected.

The implementation of ECC in the server is Single Error Correction, Double Error Detection (SECDED), as the name points, ECC can correct one bit-errors and detect two bit-errors. However, if we have more than two errors in a word, the error could occur as Silent Data Corruption (SDC) which are not detected by the ECC and may silently propagate to the application execution. To be able to detect those, we compare the output of each execution with a golden reference output of an execution when DRAM was operated at the nominal conditions for the Rodinia benchmarks, while for the CloudSuite, we implement a similar approach on the client side, validating the received data compared to a golden output. For the micro-benchmarks, we can measure the exact number of errors that manifested as SDCs.

Our results indicate that SECDED ECC that is anyway available in server-grade DRAMs, can help maintain the DRAM reliability high even under relaxed refresh rate and voltage and help to address the varying weak cells across applications. In fact, in all our experiments with all the benchmarks discussed above, we discovered only one application which trigger an uncorrectable error on a specific set of DIMMs and high temperature (70°C) close to the maximum allowed by the DRAM DIMM, while all other reported errors are only single bit-errors, which were all corrected by ECC, even when DIMMs were operated under a high temperature, up to 60°C. Note that we also have not discovered any SDCs when running the benchmarks. Nonetheless, there is a possibility that a specific combination of DIMMs, temperature and benchmarks could trigger uncorrectable errors and system failures.

© 2018. UniServer Consortium Partners. All rights reserved

4. Conclusions and Future Research

This deliverable presents the results of memory subsystem characterization for the APM's X-Gene 2 micro-server operating under scaled voltage and refresh rate parameters. Firstly, we present two new studies concerning development of efficient cache and pipeline micro-viruses to identify safe operation margins for the server and a statistical analysis that aims to predict safe voltage operation limits for the ARMv8 cores. Secondly, we demonstrate a framework to identify system level parameters which may affect DRAM reliability when running selected applications. By building the correlation between intrinsic and extrinsic parameters of the system and DRAM error behavior, we identify program inherent features which have a significant impact on errors in DRAM operating under scaled supply voltage and refresh rate. Finally, we explore the necessary testing duration to ensure that DRAM operates reliably under discovered supply voltage and refresh rate levels.

The output of this deliverable provides the necessary inputs and insights for WP4 to build the prediction model which will be presented in D4.8.

The next steps on the task T3.3 regarding the characterization of memory subsystem include:

- Finalization of the unified characterization framework on D3.7 "Final Evaluation of Cores, Caches and Dynamic Memories".
- Experiments with the rest of applications provided by the project partners.
- Extension of the system level parameter list with new parameters which may affect DRAM and chip reliability.

In the final year of the project, we will continue the characterization process using different chips and apply the same characterization routines to the X-Gene 3 micro-server, which contains the latest generation of 64-bit ARM cores and DDR4 memories.

© 2018. UniServer Consortium Partners. All rights reserved

5. References

[1] G. Papadimitriou, A. Chatzidimitriou, M. Kaliorakis, Y. Vastakis, D. Gizopoulos, "Micro-Viruses for Fast System-Level Voltage Margins Characterization in Multicore CPUs," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2018.

[2] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, D. Gizopoulos, P. Lawthers, S. Das, "Harnessing Voltage Margins for Energy Efficiency in Multicore CPUs," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017.

[3] R. J. Riedlinger, R. Bhatia, L. Biro, B. Bowhill, E. Fetzer, P. Gronowski, T. Grutkowski, "A 32nm 3.1 Billion Transistor 12-Wide-Issue Itanium® Processor for Mission-Critical Servers," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2011.

[4] M. Kaliorakis, A. Chatzidimitriou, G. Papadimitriou, D. Gizopoulos, "Statistical Analysis of Multicore CPUs Operation in Scaled Voltage Conditions," in *IEEE Computer Architecture Letters*, 2018.

[5] F. Pedregosa, et al., "Scikit-learn: Machine learning in Python," in *Machine Learning Research, vol. 12, pp. 2825-2830*, 2011.

[6] J. R. Lackritz, "Exact p Values for F and t Tests," in *The American Statistician, vol. 38, pp. 312-314*, 1984.

[7] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T.W. Keller, "Energy management for commercial servers," in *Computer Journal, Volume 36, Issue 12*, December 2003.

[8] Atwood, G., "Current and emerging memory technology landscape," in *Flash Memory Summit*.

[9] U. Kang, H. Soo Yu, C. Park, H. Zheng, J. Halbert, K. Bains, S. Jang, and J.S. Cho, "Co-architecting controllers and dram to enhance dram process scaling".

[10] Hong, S., "Memory technology trend and future challenges," in *International Electron Devices Meeting (IEDM)*, 2010.

[11] Kim, K., "Future memory technology: challenges and opportunities," in *International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA)*, 2008.

[12] J. A. Mandelman, R.H. Dennard, G.B. Bronner, J.K. DeBrosse, R. Divakaruni, Y. Li, and C.J. Radens, "Challenges and future directions for the scaling of dynamic random-access memory (dram)," in *IBM Journal of Research and Development, Volume 46, Issue 2-3*, 2002.

[13] O. Mutlu, and L. Subramanian, "Research problems and opportunities in memory systems," in *International Journal in Supercomputing Frontiers and Innovations, Volume 1, Issue 3*, 2014.

[14] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-aware intelligent dram refresh," in *39th International Symposium on Computer Architecture (ISCA)*, 2012.

[15] Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," in *In Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA '13)*, 2013.

[16] Samira Khan, Donghyuk Lee, Yoongu Kim, Alaa R. Alameldeen, Chris Wilkerson,and Onur Mutlu, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," in *SIGMETRICS Perform. Eval. Rev. 42*, 2014.

[17] Matthias Jung, Deepak M. Mathew, Carl Christian Rheinländer, Christian Weis, Norbert Wehn, "A Platform to Analyze DDR3 DRAM's Power and Retention Time," in *IEEE Design & Test 34(4): 52-59*, 2017.

[18] Tapti Palit, Yongming Shen, and Michael Ferdman, "Demystifying Cloud Benchmarking," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2016.

[19] Fitzpatrick, Brad, "Distributed Caching with Memcached," in *Linux J. 2004*, 2004.

[20] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica, "Apache Spark: A Unifed Engine for Big Data Processing," in *Commun. ACM 59, 11 (Oct. 2016)*, 2016.

[21] David Smiley, Eric Pugh, Kranti Parisa, and Matt Mitchell, "Apache Solr enterprise search server," in *Packt Publishing Ltd.*, 2015.

[22] Micron Technology, "DDR3 SDRAM UDIMM - MT18JSF1G72AZ-1G9 - 8G," 2015.

© 2018. UniServer Consortium Partners. All rights reserved

[23] Prashant J. Nair, Dae-Hyun Kim, and Moinuddin K. Qureshi, "ArchShield: Architectural Framework for Assisting DRAM Scaling by Tolerating High Error Rates.," in *In Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA '13).*, 2013.

[24] Matthias Jung, Éder Zulian, Deepak M. Mathew, Matthias Herrmann, Christian Brugger, Christian Weis, and Norbert Wehn, "Omitting Refresh: A Case Study for Commodity and Wide I/O DRAMs," in *In Proceedings of the 2015 International Symposium on Memory Systems (MEMSYS '15)*, 2015.

[25] Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu, "RAIDR: Retention Aware Intelligent DRAM Refresh," in *In Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA '12)*, 2012.

[26] R. K. Venkatesan, S. Herr, and E. Rotenberg, "Retention-aware placement in DRAM (RAPID): software methods for quasi-non-volatile DRAM," in *In The Twelfth International Symposium on High-Performance Computer Architecture*, 2006.

[27] Chung-Hsiang Lin, De-Yu Shen, Yi-Jung Chen, Chia-Lin Yang, and ChengYuan Michael Wang, "SECRET: A Selective Error Correction Framework for Refresh Energy Reduction in DRAMs," in *ACM Trans. Archit. Code Optim. 12, 2*, 2015.

[28] Samira Khan, Donghyuk Lee, Yoongu Kim, Alaa R. Alameldeen, Chris Wilkerson, and Onur Mutlu, "The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study," in *SIGMETRICS Perform. Eval. Rev. 42, 1*, 2014.

[29] Kevin K. Chang, A. Giray Yaălikçi, Saugata Ghose, Aditya Agrawal, Niladrish Chatterjee, Abhijith Kashyap, Donghyuk Lee, Mike O'Connor, Hasan Hassan, and Onur Mutlu, "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms," in *Proc. ACM Meas. Anal. Comput. Syst. 1, 1, Article 10 (June 2017), 4*, 2017.

[END OF DOCUMENT]

© 2018. UniServer Consortium Partners. All rights reserved