University of Cyprus

DEPARTMENT OF COMPUTER SCIENCE

Total Cost of Ownership Optimization for Edge and Cloud Data-Centers

Panagiota Nikolaou

A Dissertation Submitted to the University of Cyprus in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

December 2019

VALIDATION PAGE

Doctoral Candidate: Panagiota Nikolaou

Doctoral Thesis Title: Total Cost of Ownership Optimization for Edge and Cloud Data-Centers

The present Doctoral Dissertation was submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy at the ***Department of Computer Science*** *and was approved on* ***December 6, 2019*** *by the members of the* ***Examination Committee****.*

Examination Committee:

Research Supervisor:_____

Yiannakis Sazeides

Committee Member:_____

George Pallis

Committee Member: _____

Marios D. Dikaiakos

Committee Member: _____

Ramon Canal

Committee Member: _____

Theocharis Theocharides

DECLARATION OF DOCTORAL CANDIDATE

The present doctoral dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy of the University of Cyprus. It is a product of original work of my own, unless otherwise mentioned through references, notes, or any other statements.

Panagiota Nikolaou

……

To the new member of our family,

Ioanna

Acknowledgements

Firstly, I would like to thank and express my sincere gratitude to my advisor, Professor Yiannakis Sazeides for the opportunity to work under his supervision and the continuous support of my Ph.D study, for his patience and motivation. His guidance helped me throughout my Ph.D. Also, the motivation that he has given me, made the work more challenging and helped me to become a better researcher.

I would like to thank, also Professor Chrysostomos Nikopoulos for his participation in this work and the great collaboration that we had all of these years.

During my Ph.D, I have spent two months at the IROC technologies for a short term scientific mission. I would like to thank Andrian Evans and Dan Alexandrescu, for this great opportunity and for hosting me at their group in 2015.

Many thanks to all my colleagues of the xi lab. Especially, thanks to Antreas Prodromou, Andreas Panteli, Panagiotis Englezakis, Zacharias Hatzilambrou, George Klokkaris, Lorena Ndreu and Marios Kleanthous for the endless discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had throughout these years.

I would like also to thank my family and friends for supporting me spiritually throughout this thesis and my life in general. Special thanks, to my mother Giannoulla, my father Nikolas and my brother Paraskevas, for their support, advice and their unconditional love all of these years. I feel very fortunate to have them in my life. Especially, I would like to thank my husband, Marios for the understanding and withstanding. Without him the entire process would have been non-achievable and intolerable. A huge thank to all my friends, especially to Anna, Maria and Margarita for being there for me, whenever needed and for whatever needed.

Finally, I would like to take the opportunity to acknowledge the support that I have received from the following EU co-funded projects: Eurocloud FP7 (2010-2013), HARPA FP7 (2013-2016) and UNISERVER H2020 (2016-2019).

"Build up your weaknesses until they become your strong points", Knute Rockne

Περίληψη

Ο αριθμός των ευφυών συσκευών που συνδέονται με το Διαδίκτυο αυξάνεται καθημερινά και σύντομα θα είναι της τάξης των δεκάδων δισεκατομμυρίων, αποτελώντας το Διαδίκτυο των πραγμάτων (IoT). Κάθε μία από αυτές τις συσκευές στέλνει δεδομένα στο Διαδίκτυο τα οποία σύντομα αναμένεται να φτάσουν τα δεκάδες exabytes. Αυτή η ταχεία αύξηση δεδομένων θα ασκήσει πρωτοφανή πίεση στην τρέχουσα υποδομή του Διαδικτύου και στα κεντρικά κέντρα δεδομένων (Cloud). Η αντιμετώπιση αυτής της επικείμενης πλημμύρας δεδομένων απαιτεί τόσο την ενίσχυση των δυνατοτήτων επεξεργασίας των τρεχόντων εξυπηρετητών όσο και την επανεξέταση του τρόπου επικοινωνίας και επεξεργασίας δεδομένων στο Διαδίκτυο. Για το σκοπό αυτό, τα τελευταία χρόνια, τα κέντρα δεδομένων έχουν αυξηθεί σε αριθμό, μέγεθος και χρήση. Μεγάλες μονάδες κέντρων δεδομένων που αποτελούνται από χιλιάδες έως δεκάδες χιλιάδες εξυπηρετητές χρησιμοποιούνται για την παροχή υπηρεσιών, όπως ηλεκτρονικού ταχυδρομείου (email), αναζήτηση ιστού, κοινωνική δικτύωση, χάρτες κ.λπ., σε δισεκατομμύρια χρήστες. Επιπλέον, τα τελευταία χρόνια έχει προκύψει η παροχή των υπηρεσιών Cloud στο Edge, πιο κοντά στους χρήστες. Μια σημαντική επίπτωση αυτών των εξελίξεων είναι η αύξηση της κατανάλωσης κόστους και ενέργειας των κέντρων δεδομένων και συνεπώς υπάρχει αυξανόμενη ανάγκη για αποτελεσματικές μεθοδολογίες και τεχνικές για τη βελτίωση του σχεδιασμού του κέντρου δεδομένων για την αποτελεσματικότερη χρήση του.

Ο κύριος στόχος αυτής της εργασίας είναι να ελαχιστοποιήσει το συνολικό κόστος ιδιοκτησίας (TCO) ενός κέντρου δεδομένων, ενώ παράλληλα να ικανοποιήσει τη ποιότητα εξυπηρέτησης διαφόρων εφαρμογών που εκτελούνται σε κάποιο κέντρο δεδομένων. Αυτό απαιτεί την ανάπτυξη μοντέλων που επιτρέπουν την εξερεύνηση του σχεδιασμού ενός κέντρου δεδομένων. Συγκεκριμένα, πρώτα προσδιορίζονται τα μετρικά που απαιτούνται για να εξεταστεί το ενδεχόμενο ελαχιστοποίησης του TCO. Αυτή η ανάλυση τονίζει την

σημαντικότητα του TCO ως μετρικό βελτιστοποίησης και προσδιορίζει επίσης τις κύριες παραμέτρους που το επηρεάζουν. Στην συνέχεια, αξιολογούμε τα πιθανά TCO κέρδη στο Edge σε σύγκριση με το Cloud, καθώς και ένα ποιο αποδοτικό σε ενέργεια Edge που χρησιμοποιεί λειτουργίες σε χαμηλότερες τάσης (voltage). Έπειτα, προτείνουμε και αναλύουμε τις επιπτώσεις στο TCO, μιας δυναμικής τεχνικής που παρακολουθεί τα μετρικά απόδοσης και καθορίζει πότε πρέπει να λειτουργήσει μια κεντρική μονάδα δεδομένων σε κανονική ή χαμηλότερη τάση. Τέλος, αναλύουμε τις συνέπειες των λαθών των κύριων μνήμων (DRAM) και των τεχνικών προστασίας τους στο TCO και προσδιορίζουμε την κατάλληλη τεχνική προστασίας που παρέχει την μεγαλύτερη εξοικονόμηση TCO χωρίς να διακυβεύεται η διαθεσιμότητα του συστήματος.

Abstract

The number of intelligent Internet-connected devices is growing rapidly and will soon be in the order of tens of billions, forming the Internet of Things (IoT). Each of these devices is pushing data to the Internet that are soon expected to reach tens of exabytes. It is expected that such data growth will put an unprecedented pressure on the current Internet infrastructure and the centralized (Cloud) Datacenters (DCs). In order to successfully deal with this imminent data flood, it is imperative to enhance the processing capabilities of the current servers. Redesigning data communication and processing across the Internet is equally important. In light of this, DCs have increased in numbers, size and utilization. Large DCs that consist of thousands to tens of thousands of servers are used to deliver services, such as e-mail, web search, social networking, maps etc., to billions of users. Additionally, a new paradigm has emerged which makes Cloud services available at the Edge. One key ramification of these developments is an increase in the cost and energy consumption of both Cloud and Edge DCs.

The aim of this thesis is to minimize the total cost of ownership (TCO) of a DC while meeting the quality of service of different workloads running in the DC. This requires the development of innovative methods and models that enable the exploration of the design space of a DC. First, a method that detects the existence of high correlations among several application parameters such as performance, power, reliability with the cost (TCO) is evaluated. This, in fact, leads to the determination of the parameters that experience high correlation with the TCO and the further exploration of them throughout this thesis. Then, there is an evaluation of the possible TCO gains of an Edge deployment compared to Cloud as well as an Edge deployment that employs under-volting operations. Furthermore, there is a proposal and analysis of the TCO implications of a dynamic technique that monitors performance counters and determines when to operate a CPU in nominal or undervolted

settings. Finally, we present an analysis of the implications of DRAM failures and DRAM protection techniques on the TCO. This determines the appropriate protection technique that provides the most TCO savings without compromising the availability of the system.

Thesis Contributions

This thesis has contributed to the following journal articles, conference and workshop papers, book chapters and tools:

Conference papers:

1. **Panagiota Nikolaou**, Yiannakis Sazeides, "Identification of an entire workload's CPU-Vmin from the n-first seconds of its execution based on performance counters", ISPASS 2020.

2. Georgios Karakonstantis, Konstantinos Tovletoglou, Lev Mukhanov, Hans Vandierendonck, Dimitrios Nikolopoulos, Peter Lawthers, Panos Koutsovasilis, Manolis Maroudas, Christos D. Antonopoulos, Christos Kalogirou, Nikolaos Bellas, Spyros Lalis, Srikumar Venugopal, Arnau Prat-Perez, Alejandro Lampropulos, Marios Kleanthous, Andreas Diavastos, Zacharias Hadjilambrou, **Panagiota Nikolaou**, Yanos Sazeides, Pedro Trancoso, George Papadimitriou, Manolis Kaliorakis, Athanasios Chatzidimitriou, Dimitris Gizopoulos and Shidhartha Das: "An Energy-Efficient and Error-Resilient Server Ecosystem Exceeding Conservative Scaling Limits", DATE 2018.

3. Nikolaos Zompakis, Michail Noltsis, Lorena Ndreu, Zacharias Hadjilambrou, Panayiotis Englezakis, **Panagiota Nikolaou**, Antoni Portero, Simone Libutti, Giuseppe Massari, Federico Sassi, Alessandro Bacchini, Chrysostomos Nicopoulos, Yiannakis Sazeides, Radim Vavrík, Martin Golasowski, Jiri Sevcík, Vít Vondrák, Francky Catthoor, William Fornaciari, Dimitrios Soudris: "HARPA: Tackling physically induced performance variability".DATE 2017: 97-102.

4. **Panagiota Nikolaou**, Yiannakis Sazeides, Lorena Ndreu, Marios Kleanthous:"Modeling the implications of DRAM failures and protection techniques on datacenter TCO", MICRO 2015: 572-584.

Journal articles:

5. **Panagiota Nikolaou**, Yiannakis Sazeides, Alejandro Lampropoulos, Denis Guilhot, Andrea Bartoli, George Papadimitriou, Athanasios Chatzidimitriou, Dimitris Gizopoulos, Konstantinos Tovletoglou, Lev Mukhanov, and Georgios Karakonstantis, "On the Evaluation of the Total-Cost-of-Ownership Trade-offs in Edge vs Cloud deployments: A Wireless-Denial-of-Service Case Study" IEEE Transactions on Sustainable Computing

(TSUSC) Special Issue on Sustainability of Fog/Edge Computing Systems, 2019, January 2019.

6. Marios Kleanthous, Yiannakis Sazeides, Emre Özer, Chrysostomos Nicopoulos, **Panagiota Nikolaou**, Zacharias Hadjilambrou: "Toward Multi-Layer Holistic Evaluation of System Designs".Computer Architecture Letters 15(1): 58-61 (2016).

Workshop papers

7. **Panagiota Nikolaou**, Yiannakis Sazeides, Antoni Portero, Radim Vavrík, Vít Vondrák: "A Methodology for Oracle Selection of Monitors and Knobs for Configuring an HPC System running a Flood Management Application". HIP3ES2017 workshop, CoRR abs/1702.07748 Workshop (2017).

8. K. Tovletoglou, C. Chalios, G. Karakonstantis, L. Mukhanov, H. Vandierendonck, D. S. Nikolopoulos, P. Koutsovasilis, M. Maroudas, C. Antonopoulos, C. Kalogirou, N. Bellas, S. Lalis, M. M. Rafique, S. Venugopal, A. Prat-Perez, A. Diavastos, Z. Hadjilambrou, **P. Nikolaou**, Y. Sazeides, P. Trancoso, G. Papadimitriou, M. Kaliorakis, A.Chatzidimitriou, and D. Gizopoulos, "An Energy-Efficient and Error-Resilient Server Ecosystem Exceeding Conservative Scaling Limits", Energy-efficient Servers for Cloud and Edge Computing Workshop (ENeSCE) co-located with HiPEAC, Stockholm, Sweden, January 2017.

9. **Panagiota Nikolaou**, Yiannakis Sazeides, Marios Kleanthous , Lorena Ndreu , "The Implications of Different DRAM Protection Techniques on Datacenter TCO", SELSE 2015.

Book chapters:

10. **Panagiota Nikolaou**, Zacharias Hadjilambrou, Panayiotis Englezakis, Lorena Ndreu, Chrysostomos Nicopoulos, Yiannakis Sazeides, Antoni Portero, Radim Vavrik, and Vit Vondrak:" Evaluating System-Level Monitors and Knobs on Real Hardware", Book Chapter in "Harnessing Performance Variability in Embedded and High-performance Many/Multi-core Platforms" by Springer International Publishing AG, part of Springer Nature 2019.

11. Nikolaos Zompakis, Michail Noltsis, **Panagiota Nikolaou**, Panayiotis Englezakis, Zacharias Hadjilambrou, Lorena Ndreu, Giuseppe Massari, Simone Libutti, Antoni Portero, Federico Sassi, Alessandro Bacchini, Chrysostomos Nicopoulos, Yiannakis Sazeides, Radim Vavrik, Martin Golasowski, Jiri Sevcik, Stepan Kuchar, Vit Vondrak, Fritsch Agnes, Hans Cappelle, Francky Catthoor, William Fornaciari, and Dimitrios Soudris: "The HARPA Approach to

Ensure Dependable Performance", Book Chapter in "Harnessing Performance Variability in Embedded and High-performance Many/Multi-core Platforms" by Springer International Publishing AG, part of Springer Nature 2019.

Public Tools:

12. AMPRA: Analyzer of TCO Implications of Memory Failures and Memory Protection http://www2.cs.ucy.ac.cy/carch/xi/ampra tco.php

Contents

List of Figures

List of Tables

Table of Acronyms

Term	Definition
TCO	Total Cost of Ownership
DC	Datacenter
IoT	Internet of Things
QoS	Quality of Service
MTTF	Mean Time to Failure
MTBF	Mean Time Between Failures
MTTR	Mean Time to Repair
FIT	Failures in Time
SDC	Silent Data Corruption
IPC	Instructions Per Cycle
MD	Micro-Datacenters
VM	Virtual Machine
SoC	System on Chip
Floreon+	FLOod REcognition On the Net
RR	Rainfall-Runoff
HD	Hydrodynamic
SMT	Multi-Threading
DVFS	Dynamic Voltage and Frequency Scaling
ECC	Error Correction Code
HPC	High Performance Computing
CAPEX	Capital expenses
OPEX	Operational expenses
MPKI	Misses per kilo Instructions
WDoS	Wireless Denial-of-Service
CE	Correctable Errors
DUE	Detectable Uncorrectable Errors
NDE	Non-Detectable Errors
SECDED	Single Error Correction-Double Error Detection
Chipkill	Single-Chip error correction and Double-Chip error detection

Chapter 1

1 Introduction

1.1 Thesis Motivation

Total Cost of Ownership (TCO) evaluation is significant for making Datacenter (DC) design and operational decisions that minimize cost. However, it is not easy to estimate TCO because it does not only account for direct costs such as server's cost but also for indirect costs, such as power, performance and reliability [1]. Thus, tools for evaluating TCO are useful to assess the benefits and drawbacks of Datacenter design choices.

Companies like Azure and Amazon, provide such TCO Tools [2][3]. Particularly, these tools make comparisons between on-premises infrastructure and hosts running services in their facilities. Their goal is to assist the service providers to choose between running in the Cloud or build their own DC. However, such tools do not consider application parameters such as performance, power, software architecture and quality of service requirements. Moreover, prior research proposed TCO tools to guide Datacenter designs [80][100][101][102][103][104][63]. However, all these tools do not allow the exploration of design parameters, such as memory design protection choices, performance and power optimization techniques, mean time to failure (MTTF) and mean time to repair for all the server components. Moreover, it remains unclear to decide whether these parameters should be included in the TCO or not. To establish if there are possible connections between these parameters and TCO, we primarily focus on the detection of high correlations among several parameters with the cost (TCO). Once this is done, the selection of parameters with high correlation with TCO can be achieved which will be addressed in detail throughout the thesis.

The power efficiency is an important parameter for both Edge and Cloud deployments, that this thesis closely investigates. Edge computing is a recently introduced approach that has the potential to ensure the sustainability and scaling of the Internet during the upcoming Internet of Things (IoT) era [6]. A number of studies, develop schemes that manage the data processing of IoT applications across distributed DCs [7][8][9]. In these studies, data are transferred from IoT sensors to local micro-DCs for pre-processing and selection of the data to forward to a centralized DC. Typically, these applications consist of sensors that collect

and send data to a processing device. Their main quality of service (QoS) requirement is the response-time and, therefore, are suited for both Cloud and Edge deployments. However, servers which have been used to run these applications can only process- within a required detection time window- data from a limited number of sensors. In fact, these servers are also oversubscribed to process data from many sensors which are likely to suffer from QoS violations. Moreover, each sensor covers a fixed area and Edge deployments have limited power budget for servers per installation. This results in having an Edge installation that may be able to support a limited number of sensors and cover a limited area. At this point, this serves to highlight a key challenge for the successful realization of Edge computing: the sensor's area coverage. Evidently, one of the most critical challenges for the successful Edge deployment is the efficient use of the limited power of an Edge installation. Exceeding the power cap of the facility is unacceptable as it triggers a disruption of power. To avoid such overloads, both Edge and Cloud deployments rely on power capping schemes that enforce power budgets of individual servers [10][11] or over ensembles [12][13]. In this regard, the use of more power efficient servers, facilitates the increase of area coverage. Thus, it is desirable for processors to operate with low voltage and/or frequency to reduce power. Through this work, we will propose an evaluation methodology to consider various metrics such as cost, QoS, area-coverage and power efficiency, using energy efficient and high-performance devices.

Of course, it has to be considered that operating at lower voltage can lead to voltage emergencies which can cause timing violations and/or memory bit flips [14][15][16]. This is unacceptable in many situations because it may lead to silent data corruption (SDC) errors, or even application or system crashes [17][18]. Assuming checkpoint-based recovery [19] during the crash, the system will be unavailable until the rollback from the latest checkpoint [19]. A number of research findings aim to prevent crashes by using performance counters to predict voltage emergencies during the execution of a workload [20][21][173][22][23]. However, these works cannot predict upfront if an application can operate at lower voltage and, thus can suffer from a large number of crashes. Hence, this work investigates the TCO benefits of a workload's voltage operation identification method during the first n-seconds of its execution.

The method employed by DC servers to protect their memory from errors is of paramount importance of today's DC with TCO implications. For instance, an attempt to lower costs by removing protection from DRAM in Google servers, running Web search application, resulted in a subset of queries returning random documents due to a memory error that could not even be detected [4]. Consequently, DC servers employ a combination of hardware and

software techniques to accomplish the desired level of availability without compromising QoS [5]. This thesis also aims to explore advanced memory error protection schemes, consider the implications of multi-bit errors, account for the performance, power and temperature implications, consider the ramifications of collocated services and account for failing module replacements and maintenance policies. Through this in-depth analysis, the identification of the best DRAM protection technique for each application and for collocated applications, in terms of TCO, is accomplished.

1.2 Thesis Scope and Contributions

The major goal of this thesis is to optimize the TCO of a DC while meeting the QoS criteria of different applications running in the DC. This requires the investigation of a number of key parameters that affect TCO, the evaluation of DCs design decision and the development of efficient prediction methodologies that prevent the unavailability occurred by operations at lower voltage.

Towards the State-of-the-Art, the contributions of this thesis are firstly to identify the key subset of parameters required to evaluate and optimize TCO. Particular emphasis is given on the selected monitors and knobs to use to configure a computing system running an application while satisfying the application's requirements, not violating any system constraints and at the same time optimize TCO. We then, use these monitors and knobs throughout this thesis to present TCO optimizations.

Secondly, we evaluate the possible gains of an Edge deployment compared to a Cloud one, using a TCO model. Moreover, we evaluate the TCO gains of Energy efficient Edge micro-servers, that operate in lower voltage compared to Edge micro-servers operating at nominal margins.

Furthermore, we develop a practical CPU-Vmin identification method that uses training data based on performance counters taken on the first n-seconds of execution from a set of benchmarks. The method is tested on a different set of benchmarks to classify workloads into V-low and V-high after the same first n-seconds of their execution. V-low workloads are the workloads that can operate at the lowest CPU-Vmin observed during the training phase, while the V-high are the workloads that require a CPU-Vmin higher than the V-low to operate correctly. We then, evaluate the CPU-Vmin identification method based on real hardware measurements and we show that, for the specific multicore CPU, we use, it can provide safe-voltage, 99.4% of the time, reduce power on average by 7.1% as compared to operation with nominal supply voltage, and provide significant TCO savings.

Finally, we investigate the implications of DRAM errors and DRAM protection techniques in the TCO. We propose, for the first time, a framework, called AMPRA, for modeling the implications of DRAM failures and DRAM error protection techniques on the TCO of a datacenter. The framework captures the effects and interactions of several key parameters including: the choice of DRAM protection technique (e.g. single vs dual channel Chipkill), device width (x4 or x8), memory size, power, failures in time (FITs) for various failure modes, the performance, power and temperature overheads of a protection technique for a given service and mixes of collocated services. We then, underline the usefulness of the proposed framework by demonstrating it through several case studies that identify the best DRAM protection technique in each case, in terms of TCO. AMPRA framework is an online publicly available tool.

1.3 Thesis Organization

The organization of the rest of the thesis is as follows:

Chapter 2: describes background information related to different market segments and their key requirements, edge and cloud computing, reliability aspects, while highlighting DRAM reliability, power challenges and total cost of ownership fundamentals. Specialized background and related work that refers specifically to each Chapter is presented in the dedicated Chapters, respectively.

Chapter 3: identifies the key subset of parameters required to evaluate and optimize cost. Particular focus is given on the selected monitors and knobs, derived applying a heuristic correlation analysis.

Chapter 4: introduces an end-to-end TCO model that investigates the benefits of running an application, at the Edge vs. the Cloud. Focus is given to show that by providing extended voltage operating points at the Edge, TCO can be beneficial.

Chapter 5: determines a methodology to select the appropriate performances counters that reveal safe CPU-Vmin for an application in order to predict and prevent voltage emergencies that arise by operating below the nominal supply voltage for power savings. Afterwards, an investigation of the trade-offs between voltage emergencies implications and power gains in the TCO is conducted.

Chapter 6: introduces the AMPRA framework developed to investigate the implications of DRAM errors and DRAM protection techniques in the TCO. Particular focus is given in the evaluation of this framework.

Chapter 7: concludes the thesis and outlines future work as a direct result of the thesis contributions.

Chapter 2

2 Background

The effectiveness of technology evolution in terms of power, performance and reliability depends on the application characteristics and the various market segments. Market segments are categorized in: mobile devices, desktop computing, servers, warehouse-scale computers and embedded computers [24]. Mobile devices usually run Web-Based and media-oriented applications. The basic requirements of such applications are power -due to the battery of the device- and cost.

Desktop Computers, are used for a great variety of applications in both Edge and Cloud deployments. Due to the high demand of this market segment, desktop computers try to optimize performance and cost, as well. On the other hand, servers are used to provide more reliable and efficient services. Due to the high computational power, servers can be used for bank account applications, web pages etc. This kind of services have various requirements in terms of performance (throughput, response time etc.), availability of the service, security and cost.

Warehouse-scale computers [4] are collections of desktop or servers connected with local area networks to act as a single larger computer. Warehouse-scale computers usually consist of tens of thousands of servers. This kind of market segment supports applications like, search, social networking, video sharing, online shopping etc. So, performance, cost, power and availability are very critical for this kind of applications.

Finally, embedded devices are used for everyday machines such as washing machines. The main concerns for this market segment are power, cost and availability.

2.1 Key Monitoring Parameters

The previous characterization of market segments with their different applications and requirements, brings the need of the appropriate metrics and tools to monitor the status of application requirements running on different multicore systems.

The performance metrics that usually used to determine how well a processor performs with a specific running application are: instructions per cycle (IPC), number of cache

misses/number of instructions, program runtime, and transactions per minute. Additionally, QoS is another metric that is used to show the degree to which an activity satisfies the customer, in terms of response time.

Power consumption can by categorized into leakage and dynamic power. Both power consumptions can be estimated with models such as Cacti [25] for caches and Micron power calculator spreadsheet for DRAMs [26]. There are also some monitoring tools that are used in real hardware to track dynamic power consumption of the running application [27][28].

The availability of the system is the probability of a system operating correctly at a given time. This metric is appropriate for many computing and cloud applications and is correlated with the reliability of the service and the individual components. It is very important to keep the availability of the system in high levels.

Finally, cost is very critical for all the market segments, emphasizing large scale computers due to the large number of servers and the higher costs consumed in these markets. TCO includes all the aforementioned parameters (power, performance, availability) and many others. Due to the criticality of this metric, several works include TCO in their analysis [29][30].

In this thesis, we tackle the modeling of power, performance, availability in the TCO and we provide optimizations for each parameter.

2.2 The Internet of Things and Cloud Computing

We are currently witnessing the incremental development of the IoT era. IoT refers to the networked interconnection of everyday objects denoted as “things”, that are used to achieve certain design goals [31]. These devices are mainly embedded systems that communicate with other devices by sending data through the Internet [31]. The number of Internet-connected devices is growing daily and will soon be in the order of tens of billions. Figure 1 depicts the increase of these Internet-connected devices throughout the last few years. In order to process the data that are sent to the Internet, large DCs have increased in number and size in all over the world. These DCs differ from the traditional hosting facilities because they consist of large-scale servers in the “cloud” with well-connected processing and storage resources, commonly referred as cloud computing [4].

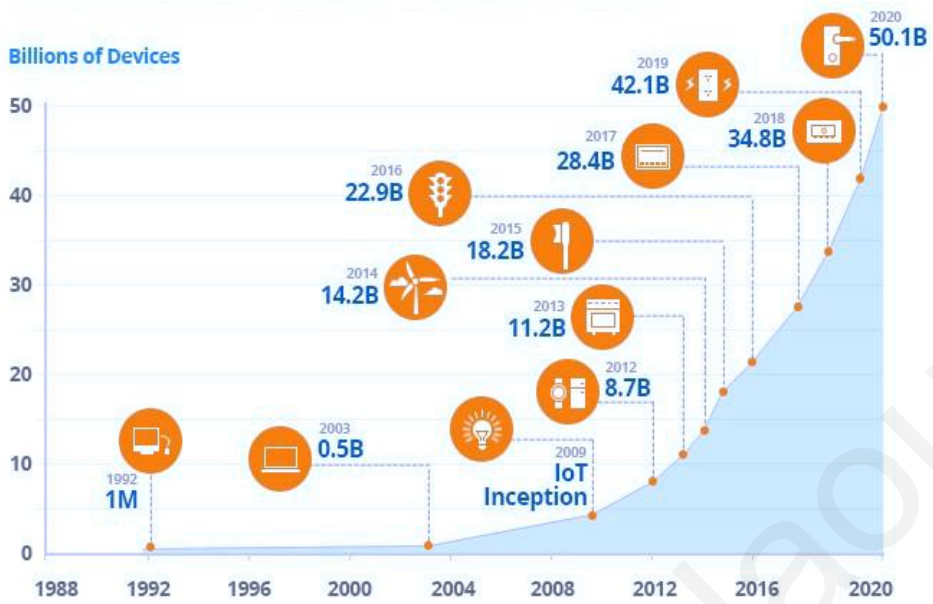

Figure 1: Internet-Connected Devices

However, the rise of cloud computing, where most compute power is located in the datacenter, comes with a number of challenges and opportunities. One of the key challenges is the communication that causes huge network traffic. In addition to this, there are some other challenges such as security [34]. Thus, it is essential to design a new architecture by considering many parameters such as reliability, scalability and QoS.

2.3 Edge Computing

The need for fast response times in various IoT applications necessitates deployments with tight QoS timing requirements. Many applications cannot tolerate latencies that exceed one or two hundred of milliseconds [135][136]. Even though Cloud Computing is centralized and requires minimal management effort or service provider's interaction, it hardly meets the QoS and response time requirements for IoT applications, due to the network latency between the sensors and a remote datacenter. On the other hand, deployments closer to the data, facilitate meeting QoS requirements by avoiding network latency [137][138]. This distributed deployment near the sources of data at many sites is referred as Edge (or Fog) computing [6]. Edge Computing is not meant to replace traditional Cloud architectures, but Cloud and Edge can work in unison to reduce the total end-to-end response time. Edge is well suited for IoT applications, where sensors collect data and send them to Edge sites for processing, thus avoiding high network latencies compared to a centralized datacenter. The Edge deployment acts as a filter that reduces the network bandwidth pressure to the Cloud [140].

According to the application scenario and the processing power of the different devices, edge computing can be based either on a two-tier [32] or on a three-tier computational model [33]. In both models the last tier corresponds to the Cloud computing resources. Moreover, the first tier includes the IoT embedded devices such as drones, sensors, devices and appliances in smart homes [32][33]. These devices need to be self-configured, self-maintained, self-repaired and make independent decisions. The main difference between the two models is that in the two-tier model, the IoT embedded devices have the computational power to process their monitored data. After processing, the data are sent to the last-tier (Cloud) to complete the processing of more complex tasks, if needed. On the other hand, on the three-tier model, the embedded devices are used only for monitoring the data. Then the data are sent to the middle-tier for processing. This tier includes different technologies such as mobile devices, normal servers or gateways and cloudlets/micro-datacenters.

Mobile devices technology: this technology includes laptops, tables and smartwatches that are located in the same facility with the IoT devices. This technology leverages idle computational power and storage space of the mobile devices to perform necessary computations [35][36][39].

Normal servers or gateways technology: analogous to mobile devices technology, this technology includes common servers that can be hosted in an Edge facility such as typical house buildings and provide computational power to the monitored data [39]. In contrast to the mobile devices, this technology provides more computational power and is dedicated and fully utilized only for the processing of the IoT monitored data.

Cloudlets or Micro-datacenters (MD) technology: in this technology each Edge deployment site can contain one or even numerous servers, called, cloudlets/MD.

Cloudlets/MD are intermediate layers that are located between the cloud and the IoT embedded devices. The only difference between the two is that in the cloudlet technology the software is provided by a cloudlet provider, in contrast to the MD that the users are responsible for the software. Due to the fact that cloudlets/(MD) are small clouds they can be referred as “datacenters in a box” [36][37][38][39]. The users can rent virtual machines (VMs) on the nearest cloudlet/MD to process their jobs [37][39][139]. The cloudlet/MD is a self-contained, secure computing environment that includes all necessary computation, storage, and networking equipment to run customer applications or applications provided by a cloudlet provider. They usually have power budget in a range of 1–100KW to meet the application demands.

There are also other models, called hierarchical, that consist of more than three tiers [8]. The theoretical comparison between flat typical models with the hierarchical shows that the second one is more beneficial in terms of latency [8].

Apart from the need to reduce the latency to satisfy an application's QoS time requirement, the communication of the data to the Cloud can lead to serious security and privacy issues, which in some cases is unacceptable to the end users [141]. Furthermore, energy efficiency and cost reduction are some other benefits of the edge technologies [39].

Figure 2, shows an IoT system architecture that includes both Edge and Cloud deployments. The Edge servers are placed near the data and are responsible for data collecting from various IoT devices, data processing and transferring a concise report to the Cloud.

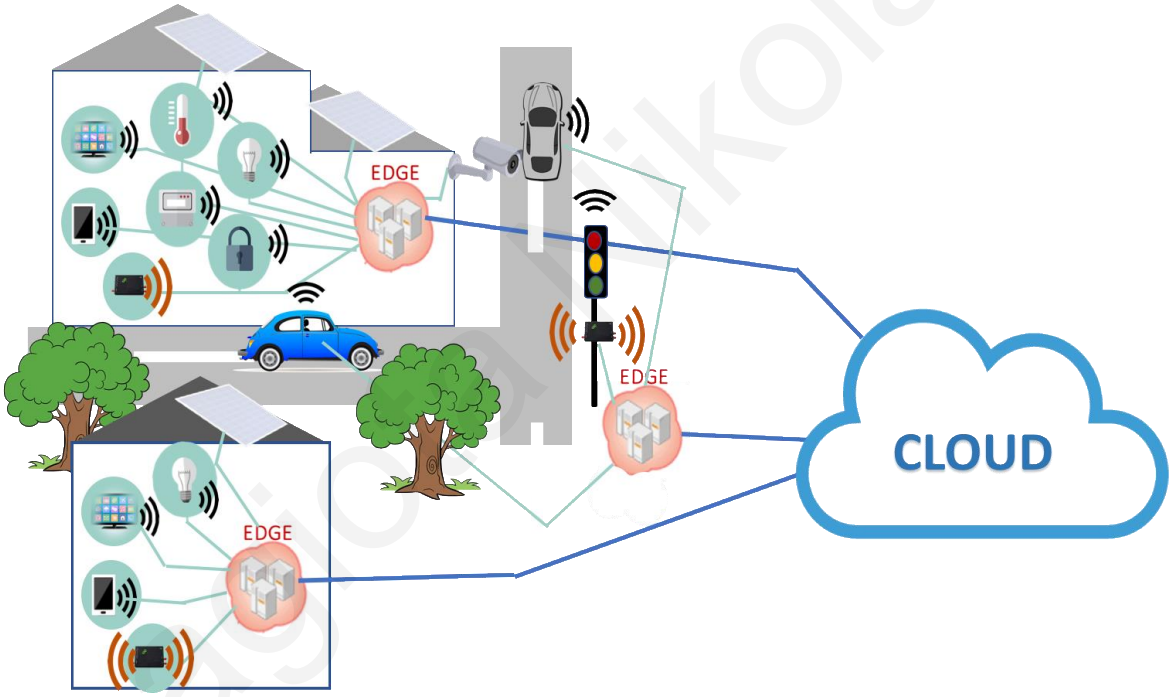

Figure 2: IoT System Architecture including Edge and central Cloud deployments

Despite the substantial advantages, Edge Computing has some limitations. A major one, is that Edge sites/facilities are power constrained [7]. Thus, the number of servers per site needs to fit the power budget that is provided by an electricity provider and is not already allocated for other uses. Edge facilities can be ordinary buildings with several other electrical appliances in use. Certainly, an electricity provider can increase the power budget at a facility but this comes with an extra cost. Consequently, Edge servers that are more power efficient may hold the key for successful Edge deployment since they will allow more servers per site and processing of data from more sensors without the extra costs to the electricity providers.

Several prior works, consider the Edge-Cloud trade-offs to decide where to place highly constrained applications and satisfy their requirements without compromising power and

availability [7][8][9][141][143][144][145][146][147]. It is also worth highlighting that existing studies of Edge deployments relied on measurements obtained from either too simple devices (e.g. raspberry-pi) or too powerful ones (e.g. classical high-end servers) which do not strike a fair balance between energy and performance that is essential in Edge installations [142][9].

Chapter 4 discusses and analyzes the trade-offs of running in the Edge or the Cloud deployments. For the Edge computational technology, the work in Chapter 4 assumes normal servers located in the same facility with the data.

2.4 Reliability Challenges

The integration of billions of transistors on a single die increases the complexity of the System on Chip (SoC). To this end more cores can be implemented on a single die. So even if the failure rate decreases per transistor, the number of transistors scales in a higher rate than the miniaturization. Along with this trend, the devices are becoming more sensitive to errors [40], and more powerful fault-tolerant techniques are needed. On the other hand, increasing reliability may sacrifice performance, power or cost.

Memory Reliability Challenges

A lot of studies indicate the importance of DRAM protection by presenting field large scale analysis [41][42]. Figure 3 shows an analysis that we did in a supercomputer with 209 nodes highlighting the importance of DRAM and cache errors. This study was based on one-year results. Figure 3 shows the faulty nodes that experience L2 cache and DRAM errors as a fraction of time. The Figure shows that L2 cache errors are distributed across several nodes and different time periods whereas DRAM errors are not. An important observation is that the total number of L2 cache errors is much lower than DRAM errors (1077 L2 cache errors, 1043041 DRAM errors). This observation motivates the investigation of more powerful protection techniques for DRAM memories. Typically, a DRAM uses a capacitor and a transistor to store a bit of data. Since the capacitor discharges very often, a refresh operation is needed to not lose the stored data [43][44]. High densities can be reached because only one transistor and one capacitor are needed to store a single bit. A DRAM is organized on a number of DIMMS, where each DIMM consists of a number of ranks. A single rank consists of multiple DRAM devices (or DRAM chips) where all or a subset of them operate together to provide 64 bits.

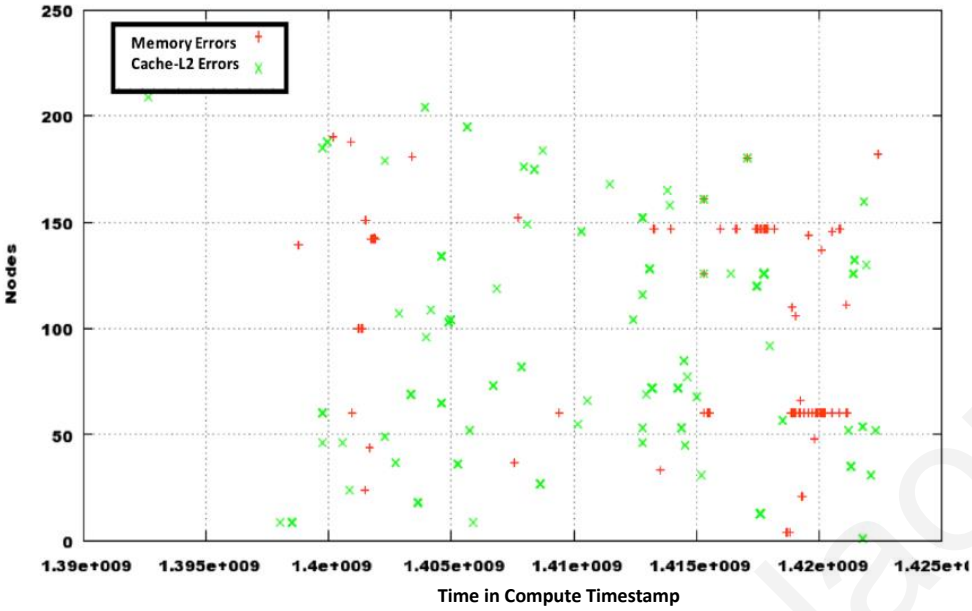

Figure 3: Faulty nodes with cache and memory errors as a fraction of time

Each device can provide 4, 8 or 16 bits (referred as x4, x8 or x16, devices respectively). For example, a 64 bit can be produced using 16 x4 DRAM devices or by using 8 x8 devices. Each DRAM device contains 8 memory arrays called banks and a multiplexer to choose 8 bits. The banks are organized into rows and columns with each cell holding one bit of data. Due to their similarity, main memory uses the same conventional protection mechanisms as caches. For example, data redundancy is used both for caches and main memory arrays. More specifically, ECC codes, Hamming [45] or Hsiao [46] can be used to detect double bit errors and to correct single bit errors in main memory. In addition, Single-Chip error correction and Double-Chip error detection or Chipkill [47], is commonly used for DRAM protection in high availability servers and large-scale systems, because it has the ability to correct all the errors that appear in a DRAM device and to detect errors in two DRAM devices. Chipkill relies on symbol-based coding to perform error detection and correction. In a symbol-based code, each codeword is composed of multiple symbols, with each representing a group of bits.

Memory reliability has been the subject of many research studies. The problem is becoming more prominent as even though the transistors scale down, the transistor count in processor still increases. Consequently, this mismatch leads to an increase in the probability of failure per component. Also, the increase of the number of memory components per processor, as long as the thermal and power headroom, increases further the probability of failure and consequently the actual failure rate. Even thought, probability of failure, per processor, is still in lower levels, future is coming with more challenges.

Moreover, reliability is closely intertwined with power and performance. For example, several proposed techniques provide high reliability by sacrificing performance or/and power. On the other hand, techniques that are not so reliable, may not hurt performance or/and power in such scale.

Chapter 6 analyzes the implications of memory errors and memory protection techniques for different applications.

2.5 Power Challenges

Another important aspect that we tackle in this thesis is the power consumption. It is one of the main parameters that affects TCO. TCO is defined by the amount of power that a server consumes and the additional power required to run the server, which includes power conversion and cooling.

Power is strictly correlated with performance (higher performance leads to more power consumption). Basically, device dynamic power depends on the capacitance of the device, voltage, activity and frequency as shown in the following equation:

$$Dynamic\ Power = C * V^2 * F * A$$

where C is the Capacitance, V is the supply Voltage, A is the Activity and F is the clock frequency. On the other hand, leakage power comes from the sub-threshold leakage and gate leakage. Subthreshold leakage happens when the gate of the transistor is off but it shows non-zero amount of current even for voltages lower than the threshold voltage (Vth) [43].

As long as the number of transistors and the transistor's frequency increase there is more power demand for both leakage and dynamic power.

The power efficiency of an application depends on both software and hardware components. There are various technologies that are used from the development of a new hardware to the use, to reduce power consumption and consequently TCO. In this thesis we are evaluating existing hardware with several well-known techniques, that reduce power consumption.

One technique that aims to provide relief from stringent power constraints is under-volting: operate a CPU at a lower than nominal voltage [164][169][170]. However, a naive approach where a CPU is always undervolted, makes the CPU more susceptible to variations, such as voltage fluctuations or voltage emergencies, which can cause timing violations or bit flips [14][15][16] which in turn may lead to SDC errors, or even application or system crashes [17][18].

Another technique to avoid power overloads in various deployments is power capping that enforce power budgets of individual servers [10][11] or over ensembles [12][13].
Chapter 5 proposes a new detection approach for identifying CPU under-volting settings to prevent system from crashes and SDC errors.

2.6 Total Cost of Ownership Fundamentals

The TCO is determined by its capital and operational expenses and is influenced by the following five main parameters:

- Infrastructure Cost: the cost of acquisition of a DC building (real estate and development), power distribution and the cooling equipment acquisition cost.
- Server Cost.
- Networking Equipment Cost.
- Operating Expenses: the cost of electricity for servers, networking equipment and cooling.
- Maintenance and Staff Expenses: the cost for repairs and the personnel salaries.

DC infrastructure, server and networking equipment costs represent the capital expenses, whereas the DC operating and maintenance costs represent the operational expenses.
The TCO is determined by the sum of capital and operational expenses. Capital expenses (CAPEX) include the cost of acquisition of a building, the power capex costs with the electricity payment, the cooling equipment acquisition cost, the cost of acquiring the servers including all their components and networking equipment costs. On the other hand, operational expenses (OPEX) include operation power and maintenance costs.
The TCO of a deployment consisting of N servers is determined as the sum of Capex Cost (C_{Capex_i}) and Opex Cost (C_{Opex_i}) of all the servers (i) as follows [80]:

$$TCO = \sum_{i}^{N} [\ C_{Capex_i} + C_{Opex_i}\ ]$$

An overview of the simple TCO framework is shown in Figure 4. For each different server configuration type (compute nodes, database nodes, storage nodes), the estimation starts with spares estimation that determines (i) the number of hot spares required to mitigate performance variability and ensure meeting performance requirement for the peak workload, and (ii) the number of cold spares needed due to server failures. The number of active servers, the initial number of servers estimated assuming no variability plus the hot spares, will determine the costs for datacenter infrastructure, server acquisition, networking equipment,

and power. The cold spares are used to determine the maintenance cost. These costs are then summed together to produce the contribution to the TCO of a given server type. The global TCO is the sum of the contribution from all server types (shown in the above equation). Prior works proposed to guide the Datacenter design, accounting TCO as the key parameter [80][100][101][102][103][104][63]. We choose TCO presented in [80] as the tool that we will extend in this thesis. This is a holistic TCO tool consisting of main parameters such as performance and power and, thus, provides more accurate TCO results than all the other available tools. In particular, throughout this thesis, we will investigate parameters that are correlated with the TCO and used to provide optimizations by extending the TCO tool in [80]. To accomplish this, we use a methodology based on correlation analysis that determines the parameters required to minimize TCO. Furthermore, to provide TCO optimizations we will develop several frameworks. The application architecture is also important and should be investigated in all the evaluations. For each of the works we will explore different applications that are related on each of the thesis objectives. Particularly we will use a high performance computing (HPC) application (FLOREON+ [48]), an IoT application (Wireless denial-of-service attack's detection application [49]), a Cloud application (Web Search [50][51]) and a lot of batch applications (Data Analytics [50][51], SPEC 2006 [52], SPEC 2017 [53], PARSEC [180] and NAS [54]). It is important to highlight that each of the frameworks that we propose is orthogonal to any other application.

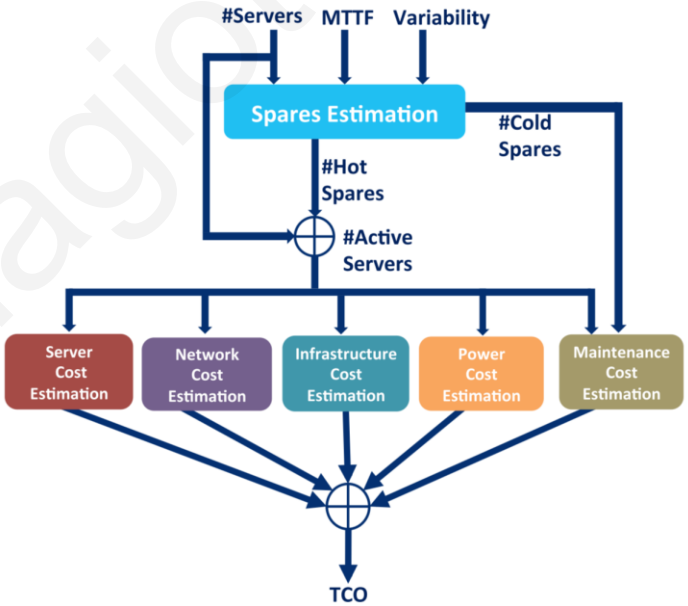

Figure 4: TCO Framework Overview [80].

Chapter 3

3 Identification of Key Subset Parameters Required to Evaluate and Optimize TCO

Various infrastructures run a diverse set of applications with different QoS requirements [5]. These requirements come in various forms, such as operational power, performance, energy, cost and availability. Naturally, computing systems need to be configured in a way to satisfy those application requirements [55]. To configure a system, all the different system hardware and software knobs, such as frequency, are set at specific settings, with many remaining at a default setting, and then, while the application runs, various monitors, such as execution time, are observed to determine whether the different requirements are satisfied. Unsurprisingly, the configuration space is extremely large and such configuration search efforts are in practice adhoc and non-optimal.

One way to reduce the configuration search dimensionality and complexity, is to reduce the requirements, monitors and knobs that need to be satisfied, observed and explored, respectively. Reduction of a problem's dimensionality is not a new problem for computing system analysis [56][57][58][59]. Such reduction, typically, relies on some form of statistical correlation, for example, principal component analysis [60][61].

In this Chapter we identify the minimum set of monitors and knobs to use for configuring a computing system that runs a specific application while satisfying the application requirements and provide cost optimizations. To accomplish this, we use a well-known correlation methodology that relies on data obtained from a detailed exploration of a configuration space.

Specifically, for this investigation we consider data obtained using eleven system monitors when exploring many settings for six knobs.

3.1 Experimental Setup-Application

We use Floreon+ application (FLOod REcognition On the Net) [48], with high QoS requirements. Floreon+ is an online system for monitoring, modeling, prediction and support disaster flood management [62]. The system focuses on acquiring and analyzing relevant data in near-real time. The data are used to provide short-term flood prediction by running hydrologic simulations.

The main processes of Floreon+ application are organized as follows:

1. Get information about actual river and reservoir situation.
2. Rainfall-Runoff (RR) modeling: simulation of surface runoff.
3. Hydrodynamic (HD) modeling: flood lake simulations, flood maps, simulations of water elevation and water velocity, a real-time hydrological model for flood prediction, water quality analysis, etc.
4. Erosion modeling: simulation of water erosion.
5. Collection and archiving of flood data that can be used to estimate the magnitude of a flood based on historical evidence.

In this Chapter we are investigating the uncertainty of the RR modeling which is the most computationally intensive part [48][62].

The application framework for the uncertainty of RR model provides an environment for running multiple simulations every repetition, when new data arrives on a system. The uncertainty contains information about how accurate is the solution that RR model provides. RR model is a dynamic mathematical model, which transforms rainfall to flow at the catchment outlet.

The uncertainty is computed as Monte Carlo samples. The Monte Carlo method gives a straightforward way of massive parallelism by increasing the number of random values working concurrently to obtain numerical results. Previous experiments [62] exhibit a good scalability of the Monte Carlo method in an HPC cluster with 64 nodes of each containing 16 cores. Figure 5 shows the normal operation of Floreon+. As Figure 5 shows, a batch of Monte Carlo iterations is running in a number of nodes (Server 1- Server n) in such a way that application's QoS requirements are satisfied. Each interval indicates the execution of a different simulation. For example, the 1st interval refers to the 1st simulation. After the execution of all the Monte Carlo iterations, the results send to a master server for processing. The total simulation time includes the execution time of Monte Carlo iterations and the time needed to process the results. When a simulation ends, the servers remain idle for a set of a period. The duration of this period is determined by the availability of the new batch of data.

Under normal operation (fault-free) the simulation always finishes within the time constraint. However, there are some cases where a fault on a component can delay the execution of the simulation, as shown in Figure 6.

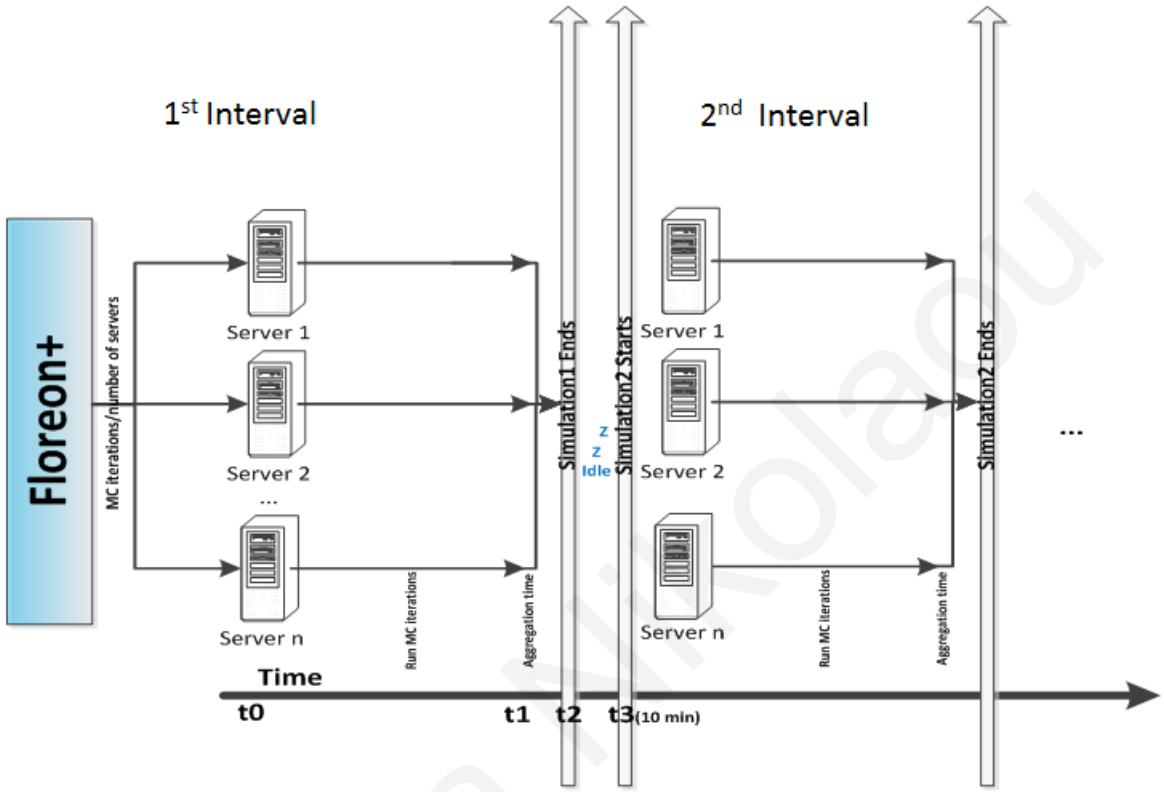

Figure 5: Floreon+ normal operation

These cases can be categorized in the following:

1. Delay the execution of the simulation but still the simulation finishes within the time constraint. The availability of the system does not decrease.
2. Delay the execution and violate the timing constraint with the same number of servers. Thus, the results of this simulation are useless and the availability of the system decreases.
3. Delay the execution and violate the timing constraint with less servers. In this case the faulty server needs to be taken offline until it is repaired or replaced. In this case the results are lost and the availability decreases.

Figure 6 illustrates the last case where the server needs to be taken offline until it is repaired or replaced. As shown in the figure, the number of servers in the 3rd interval is decreased and the job is assigned to the remaining servers until the faulty server is repaired or replaced.

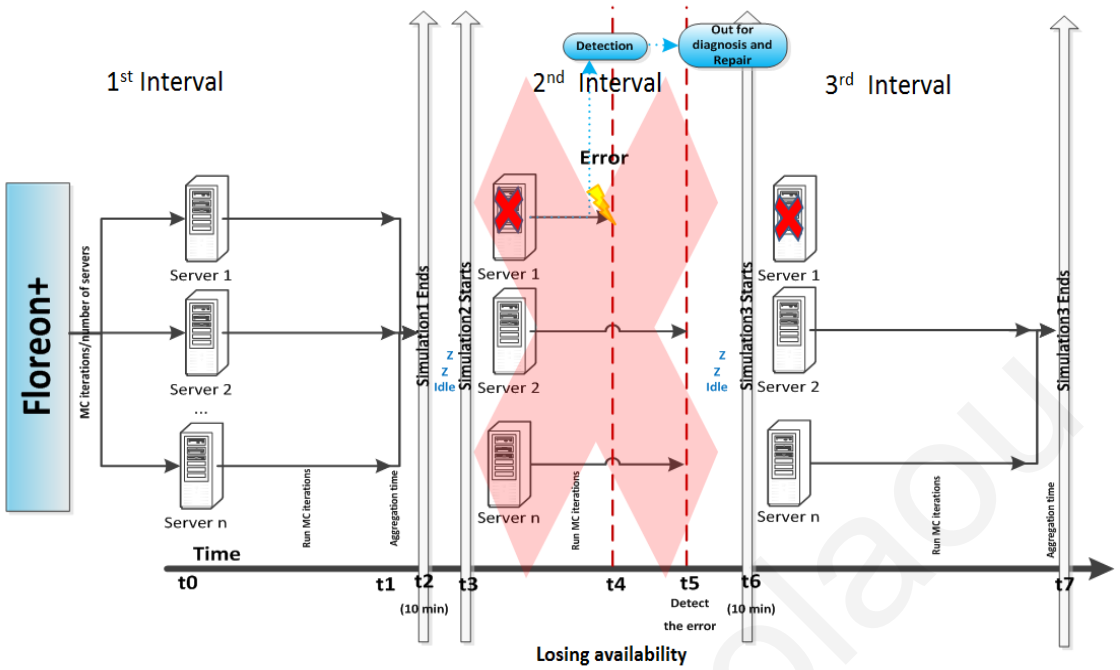

Figure 6: Floreon+ operation with faults

3.1.1 Application Requirements

Floreon+ has two running operation modes, the standard operation mode and the emergency operation mode. Both have different requirements. Standard operation mode is the default operation of the system. In this operation the weather is favorable and the flood warning level is below of the critical threshold.

On the other hand, on the emergency operation mode the water in the rivers rises due to continuous rain or free-flowing streams that are created due to heavy rainfall on small areas. During this operation mode much more accurate and frequent simulation computations are needed and the results should be provided as soon as possible. In this work we focus on the emergency operation which has tighter timing requirements and consequently we will highlight better the correlation.

3.1.2 Availability and QoS Requirements

The reliability and availability target of Floreon+ running on emergency operation is accomplished through a combination of hardware/software mechanisms and policies. This also aims at satisfying the QoS requirements even in the presence of errors and offline servers. These mechanisms typically rely on hardware and software monitors and knobs.

In general, when a server fails and if its repair time is expected to be long, the system software migrates the failed job to another server. The failure of the server is detected by a hardware or software monitor. The migration is possible because for QoS and availability reasons. Usually, computing systems are over-provisioned with spares for dealing with errors and offline servers. Server over-provisioning is determined by the availability of a system. The less available system, the more servers needed [63].

Because of its significance, emergency operation requires responsiveness in 10 minutes for each simulation. Also, it must provide high levels of availability, two nines (0.99), which may require over-provisioning with extra servers to deal with various hardware failures.

Floreon+ and other offline services can run together (collocated) to improve utilization [64][65]. Specifically, when Floreon+ satisfies the QoS requirements without using all the available cores in a server, the remaining cores can run other services. This must be done without affecting the QoS of Floreon+ and violating its requirements. Since we are going to explore the emergency operation, Floreon+ is running in isolation, utilizing all the available resources without any other service concurrently running on the same server.

3.1.3 Accuracy on Monte Carlo Iterations

It is of utmost importance that the results are as precise as they can. The precision of the simulated results is based on the number of Monte Carlo iterations [66]. It has been shown in [66] that the number of iterations has to be in the order of 10^4 to 10^5 to obtain a satisfying precision. In this work, we assume 20000 Monte Carlo iterations that has to be computed before the deadline (i.e. ten minutes, since new input data arrives from weather stations) as the baseline configuration.

Table 1 summarizes the specific values for the requirements of the Floreon+ application.

Table 1: Floreon+ Requirements

Performance	Simulation $\leq$ 10 minutes
Accuracy	$\geq 10^4$ MC iterations
Availability	≥ 0.99
Power	$\leq$ 81 Watts
Energy	$\leq$ 48600 Joules

3.2 Available Monitors and Knobs in Real Systems

A large number of monitors and knobs exist in real systems. Monitors enable the observation of physical, micro-architectural, and operating system phenomena that can assess the status of a system as well as the progress towards completing a task. On the other hand, knobs enable the proactive or reactive control of various phenomena.

Monitors and knobs in real systems can be categorized into the following categories depending on what metric they influence: performance, power, temperature and reliability.

Table 2 shows the monitors and knobs that are going to be explored in this work. This subset is by no means comprehensive and future work should consider a larger set.

Table 2: Monitors and Knobs list

Monitors	Knobs
Execution Time	DVFS
Instructions per Cycle (IPC)	SMT
DRAM Power	DRAM Protection
CPU Power	Turbo Mode
Peak Power	Prefetchers
CPU Temperature	Redundancy
Misses per Kilo Instructions (MPKI)	
Server MTBF	
System MTBF	
Capex Expenses	
Opex Expenses	

Monitors that are going to be investigated are: execution time, Instructions per Cycle (IPC), Misses per kilo Instructions (MPKI), DRAM, CPU and peak power and CPU temperature. Also, Mean Time Between Failures (MTBF) per server and for the whole system is used through analytical models and Failures in Time (FIT) rates [41]. Finally, CAPEX and OPEX expenses are going to be estimated based on publicly available info, e.g. list prices, and runtime measurements. CAPEX expenses include infrastructure, server and networking equipment costs, whereas OPEX expenses include power and maintenance costs.

Table 2, also shows the different knobs that we are going to experiment with. As the table shows, we use Simultaneous Multi-Threading (SMT) [67], Dynamic Voltage and Frequency Scaling (DVFS) [68], data prefetchers [69] and Intel's Turbo Mode [70]. Also, this work provides results with and without redundant cores. Redundant cores are used to improve the reliability of the system by migrating the running thread of a faulty core to a spare [71]. For the redundancy scenario it is assumed that half of the cores remain idle to provide higher availability. On the other hand, in the scenario without redundancy all the available physical

resources are utilized. Furthermore, this study explores the implication of using two different DRAM Protection Techniques (No Protection or ChipkillDC).

DRAM is protected from errors by using extra devices per DIMM to store Error Correction Code (ECC) bits. Modern processors usually support Chipkill with 16 ECC bits to protect 128 data bits that are interleaved across two DIMMs placed in two channels [72][73][74]. This is referred as ChipkillDC or Lockstep where it can correct all the errors in a single device and detects all the errors in two devices [72]. Chipkill can waste bandwidth, hurt performance and increase energy consumption [75][76][77]. On the other hand, No Protection does not provide any protection on DRAM.

3.3 Experimental Framework and Correlation Analysis

3.3.1 Experimental Framework

For evaluation, we use a cluster with dual socket Intel Xeon E5-2640 v3 system configuration, as shown in Table 3.

Table 3: Server Configuration

Number of CPUs	2
CPU	Intel Xeon E5-2640 v3
Number of cores per CPU	8
Number of threads per core	2
Channels per CPU	4
DIMMs/channel	2
DIMM capacity	16GB

We run each experiment 5 times, and each time we monitored all the monitors presented in Table 2. The results presented are calculated by removing minimum and maximum values and calculating the average.

To change the knobs, prefetchers and DRAM protection techniques we access BIOS, through a BIOS Serial Command Console interface (CLI) [78].

Our evaluation used Floreon+, an HPC application with a dataset of 44KB. This is a representative dataset size for the application purposes, that is used in reality, because it uses five days observations to provide predictions for the next two days.

All the power numbers are collected using the Likwid-powermeter [28] which allows monitoring the power consumption of CPU and DRAM at any given time. The results are used to calculate total power and peak power numbers.

To track CPU temperature, we use lm-sensors [79]. To estimate Server MTBF and System MTBF monitoring values, as well as, availability values we use different analytical models based on binomial probabilities.

Also, to estimate CAPEX, OPEX expenses as well as total cost we use COST-ET and AMPRA tools proposed in [80][63].

For a baseline configuration, we select the one that is currently used to run this application and includes the following values for each parameter: SMT: OFF, Frequency: 2.6 GHz, DRAM Protection: No Protection, Turbo Mode: Enable, Redundancy: 0 (No), Prefetchers: ON.

The data used for correlation analysis are obtained by exploring the 128 combinations of knobs presented in Table 4. For each configuration combination the eleven monitor values are recorded.

Table 4: Values of Knobs

Knobs	Value
DVFS	1.2, 1.7, 2.2, 2.6 (GHz)
SMT	Disable, Enable
DRAM Protection	No Protection, ChipkillDC
Turbo Mode	Disable, Enable
Prefetchers	Disable, Enable
Redundancy	Disable, Enable

3.3.2 Identification of the Key Subset of Parameters

The analysis to reduce the number of monitors and knobs that are correlated with the TCO is described below. The data that drive this analysis are obtained as described in the previous Section. The correlation analysis is done using the R statistical language [81].

The methodology used, is as follows:

1. For a given monitor, we compute the correlation coefficient (using Pearson correlation analysis) with all the other monitors. For each pair (x_i ,x_j) of monitors i and j, where $i \neq j$, that exhibit significant correlation coefficient (above a 90% threshold, the specific threshold is picked from empirical analysis), we check which of the two monitors can be removed. The monitor that shows smaller correlation coefficient with all the other monitors is removed from the list. This process is iterated over all remaining monitors.

2. Furthermore, we compute the correlation coefficient between the TCO and all remaining monitors and select the monitors with the highest correlation above a 70% threshold.
3. For all the remaining monitors and all the available knobs, we compute the correlation coefficient between them and knobs that have a correlation coefficient above a 40% threshold (the specific threshold is picked from empirical analysis) are kept.

This analysis aims to reduce the number of configurations that need to be explored to determine the configuration that provides the highest satisfaction of the TCO according to the ranking described above. Specifically, the analysis returns a subset of the monitors and knobs. All possible configuration combinations are then evaluated for the selected knobs. For the knobs that are not selected we used the baseline configuration values. Afterwards, the selected configurations are sorted according to the selected monitor(s) value(s) (if there is more than one monitor, equal weighting is used to combine them). The top ranked configuration using the selected knobs and monitors is then compared with the configuration that considers all. Their difference is measured as the maximum negative percentage difference for any of the requirements (if it is 0 for all it means it matches the best possible configuration).

3.4 Results

3.4.1 Correlation Between Monitors, Knobs and Cost

Our analysis reveals that OPEX cost is strictly correlated with CPU power and thus we removed OPEX from the list of monitors and the presented results.

Figure 7 shows the results of correlation analysis between TCO and the explored monitors. X-axis presents the list of monitors except OPEX, whereas the y-axis presents the correlation coefficient between monitors and cost (TCO). The correlation coefficient ranges from -1 to +1. A value closer to +1 means that this monitor has almost a linear relation with the TCO. A value closer to -1 means that this monitor has an inverse relation with the TCO. A value closer to 0 means that there is no correlation between the monitor and TCO.

As we can see from Figure 7, TCO can be monitored by Execution Time, Server MTBF, CPU Power and Capex cost, because they experience correlation above 0.7. The most correlated monitor is the CAPEX cost. CAPEX cost consists mainly of the number of servers

that are needed for a datacenter. These servers can be the initial servers, hot spares and cold spares.

Figure 7: Correlation analysis, showing the correlation coefficient between Monitors and TCO

Finally, Figure 8 shows, the correlation between knobs and monitors. X-axis presents the remaining list of knobs for each monitor, whereas the y-axis presents the correlation coefficient between knobs and monitors. As we can see from the Figure, DVFS, SMT, DRAM Protection and Redundancy are the selected knobs. On the other hand, Turbo Mode and Data prefetching can be reduced from the search space.

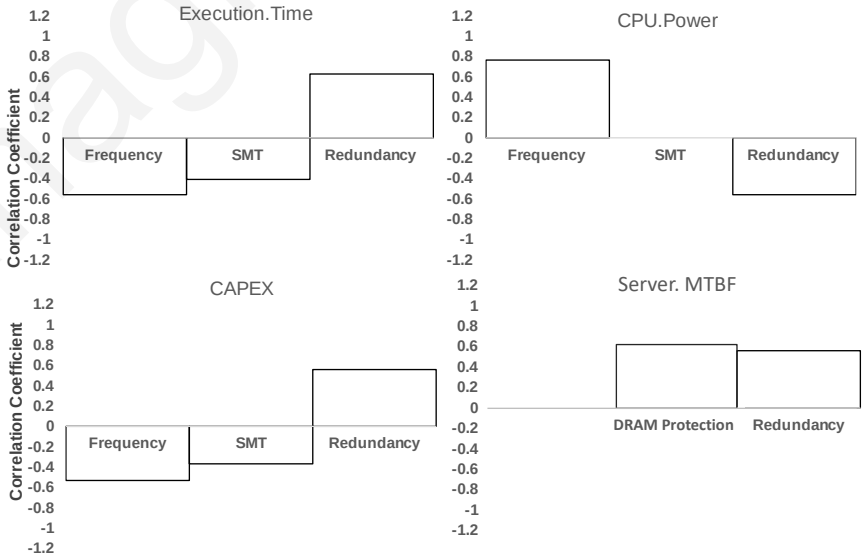

Figure 8: Correlation analysis, showing the correlation coefficient between Monitors and Knobs

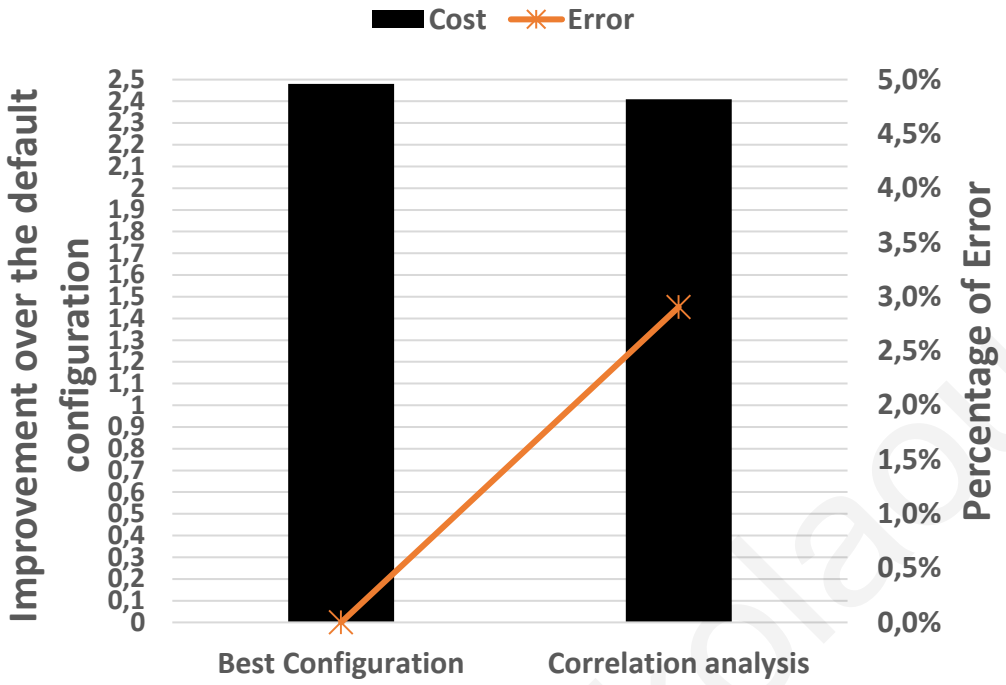

Figure 9: Improvement among the default configuration, correlation analysis predicted configuration and the best configuration

3.4.2 Validation of the Selected Monitors and Knobs

Figure 9 validates the selected monitors and knobs by evaluating the best configuration that includes all the monitors and knobs with the subset of monitors and knobs revealed from the correlation analysis.

This graph, shows the normalized with the default configuration (the initial configuration of the system) TCO improvement between the best configuration and the selected monitors and knobs revealed from the correlation analysis. As can be seen from Figure 9, TCO can be improved by changing the system configuration, by 2.4x tines compared to the default configuration. Moreover, the Figure shows that the results based on the correlation analysis are very close to the results with the best configuration. The error among the two is around 3%. This indicates that the selected monitors and knobs are the appropriate for TCO estimations and optimizations. All the selected monitors and knobs are used for further exploration in the following Chapters.

Chapter 4

4 Investigation of the TCO Benefits of Running IoT Applications at the Edge vs. the Cloud

Edge/Fog computing is a recently introduced approach that has the potential to ensure the sustainability and scaling of the Internet in the IoT era. This paradigm advocates for the execution of services closer to the sources of data [6][132], aiming this way to reduce application latency between the end user and the datacenter and at the same time relaxes the pressure on network bandwidth. Figure 10, shows the cumulative distribution of the end-to-end latency when a specific application runs in Edge and Cloud deployments.

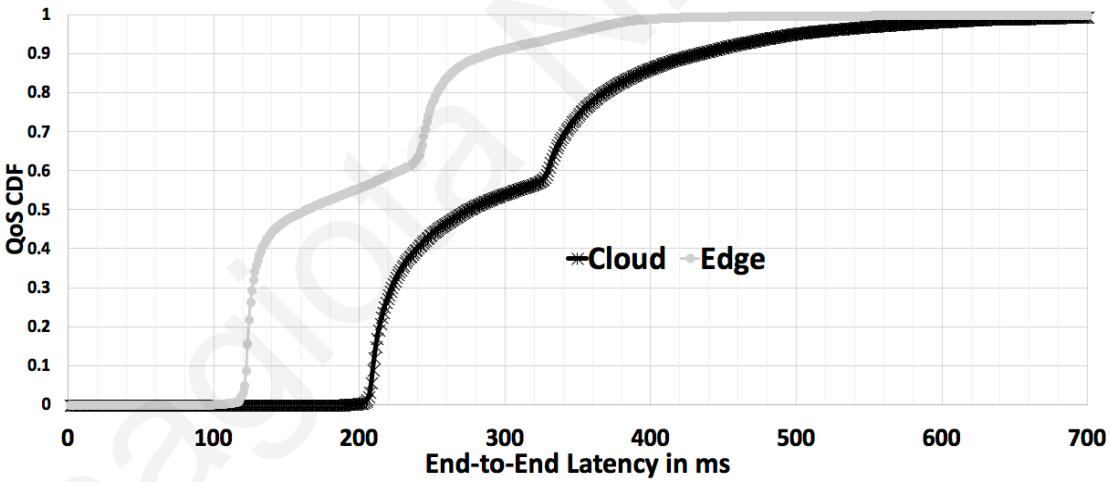

Figure 10: End-to-End Latency for running the application in Edge and Cloud deployments

The end-to-end latency includes the network and compute time of the application (details about the methodology used to obtain these results are given in Section 4.4). Figure 10 reveals considerable difference between the End-to-End latency for the Cloud and Edge deployments. As can be seen, this difference may be as large as 100 milliseconds. For a high QoS response time constraint application this extra 100 milliseconds latency may render infeasible to run the application on the Cloud or in the best case requires an expensive Cloud deployment to ensure fast processing latency.

Recent studies develop schemes that manage the data processing of IoT applications across distributed datacenters [7][8][9]. In these studies, data are transferred from IoT sensors to local micro-datacenters for pre-processing and selection which of the data to forward to a centralized datacenter. Examples of IoT applications with a tight response time and QoS constraints include face recognition [146][147], traffic counting and video processing applications [168], as well as, applications for detecting jamming attacks of wireless networks [133] [160]. All these applications consist of sensors that collect and send data to a processing device. Their main QoS requirement is the response-time and, therefore, are naturally suited for Edge deployments. However, servers used to run these applications can only process, within a required detection time window, data from a limited number of sensors, or put in another way, servers oversubscribed to process data from many sensors will suffer from QoS violations. Moreover, each sensor covers a fixed area and Edge deployments have limited power budget for servers per installation. Consequently, an Edge installation may be able to support a limited number of sensors and cover a limited area. This highlights a key challenge for the successful realization of Edge computing: the area covered by the sensors. Evidently, the most critical challenge for the successful Edge deployment is the efficient use of the limited power of an Edge installation. Exceeding the power cap of the facility is unacceptable as there will be a disruption of power. To avoid such overloads, both Edge and Cloud deployments rely on power capping schemes that enforce power budgets of individual servers [10] [11] or over ensembles [12][13]. In this regard, the use of more power efficient servers, facilitates the increase of area coverage without exceeding the Edge's or Cloud's power budget.

In this Chapter, we characterize an IoT application with tight response time QoS requirements using a state-of the art 64-bit ARMv8 based micro-server. Such a server is an excellent representative of the high-performance devices based on energy efficient-embedded devices that are required to support Cloud services at the Edge without complex cooling and power supply infrastructures. We evaluate the trade-offs among the area coverage, power efficiency and QoS when running the applications in an Edge vs a Cloud environment. To accomplish this, we rely on a new metric: the **Total Cost of Ownership (TCO) over area coverage.**

To the best of our knowledge, this is the first work that provides a holistic evaluation that considers different metrics, such as TCO, QoS, area-coverage and power efficiency, using an energy efficient and high-performance device.

4.1 Background and Challenges

4.1.1 Wireless Denial of Service Application

The application that we evaluate is a Wireless Denial of Service (WDoS) attack's detection application [49]. Current wireless networks are vulnerable to attacks by devices readily available in the market [134]. Such devices can essentially jam a wireless network and thus disrupt any running application. The WDoS application processes data, sent by sensors that continuously scan the wireless spectrum and, with the assistance of signal processing algorithms and filters, detects jamming attacks. WDoS prevention applications can detect jamming attacks and increase the availability of secure and resilient wireless networks used to connect the IoT devices at the Edge.

Wireless networking plays an important role in achieving ubiquitous computing where network devices are embedded in environments that provide continuous connectivity and services, thus improving human's quality of life.

However, due to the exposed nature of wireless links, current wireless networks can easily be attacked by jamming technology. Jammer detectors are commercially available as countermeasures against jamming systems [148][149][150][151][152][153][154] [155][156] [157][158][159][160][161][49]. In this work, we evaluate one of these WDoS solutions [49].

WDoS is a standalone solution that monitors the entire wireless spectrum using various sensors to detect anomalies derived from a Denial of Service attack which renders all the wireless devices useless. This solution does not need to be integrated internally in the wireless network, and offers a wide and easy-to-deploy solution for the most heterogeneous and challenging critical infrastructure wireless environments. To that end, the WDoS firstly performs a detailed analysis of the radio frequency spectrum, and then processes the acquired data to identify potential anomalies, giving rise to alarms and warning messages.

The WDoS solution uses several sensors with antennas connected to a Software Defined Radio (SDR) module which digitalizes the radio spectrum to a binary stream and transmits this to a processing board. The board processes in real time the incoming data while applying different filters and algorithms to match the signals found to four types of well-known jamming signals: Pulsed Jammer, Wide Band Jammer, Continuous Wave Jammer and LFM Chirp Jammer. When the WDoS application detects an attack, this incident is communicated by the processing board to the monitoring server that runs on a separate machine. Finally, a

visualization tool visualizes all the attacks reported to the monitoring server in real time. The main architecture of the solution is shown in Figure 11.

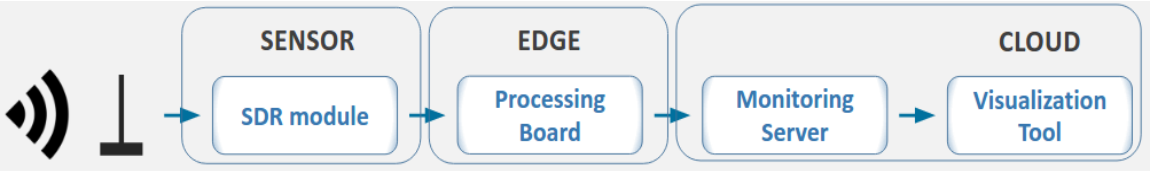

Figure 11: Architecture of DoS Jammer Detector Application including Edge and Cloud Deployments

Powerful Edge servers would allow the processing board to be simple and low cost, as it can concentrate the processing of high amounts of radio frequency spectrum data at the Edge. With powerful Edge servers, more than one instances of the processing application can be executed on the processing board (Edge server) by connecting various sensors on it and reducing the number of processing boards. The jammer detection results can be transmitted to the Cloud for storage, visualization, and post-processing.

4.1.2 WDoS Application Requirements

The WDoS application has several requirements in terms of availability, timing QoS, data transition ratio and sensor's area coverage.

4.1.2.1 Availability Requirements

Availability greatly depends on the type of installation in which the solution is deployed. Some of the most demanding installations require 99% availability of the attack detection service, i.e. the service should be available in 99% of the total service time. For some installations, such as smart construction service monitoring deployments, the availability is not so critical because of a low level of criticality of a service or a small amount of data transferred for the processing, which could be a few bytes per hour or even less. Availability of 50% or less should be optimal for shopping malls or train stations, where the wireless network is used by users for non-critical purposes, such as recreational activities.

In this work, we consider the high availability requirement of 99% to evaluate a highly constraint deployment of the application.

4.1.2.2 Quality of Service (End-to-End Latency) Requirements

We measure the WDoS latency as the time it takes to detect an attack. This time is a function of the width of the band that is analyzed. The QoS for the detection time is 400 milliseconds for the 90% of the decisions (if it is jammer detection or not) on a 5 MHz band. This is the end-to-end time that includes both the transmission time of the data to the processing board, as well as, the compute time for processing the data. Thus, QoS is determined by the sum of the processing time and the data transmission time over the network. Overall, a high-performance computing server may help to reduce the detection latency but this comes at a cost of increased energy consumption.

4.1.2.3 Data Transmission Rate

The highest data rates exist between the SDR module and the Processing Board. In this case, the maximum rate is 305 Mbps (5 Msps) and the lower rate is 30.5 Mbps (0.5 Msps). Higher data rates enable better detection accuracy, however lower rates can be also useful. The data transmission rate between the processing board and the monitoring server is much lower, as the processing board transfers only about detected attacks, i.e. the type of a jammer attack, frequency, jammer power and a timestamp. This will result in about 100 bytes of payload per packet and a minimum of 40 packets per second (one packet per decision per algorithm). To assess the benefits of Cloud deployment, we use the lowest data transmission rate, imposing an assumption that there is no any bandwidth degradation.

4.1.2.4 Sensor's Area Coverage

Each sensor that monitors the wireless band of 2.4GHz covers an area of approximately 25 square meters. Thus, in real deployments, several sensors should be used together to cover a large area, such as a shopping center floor or an airport security screening area. For such areas, multiple instances of the WDoS application should be run on the same processing board. However, this might stress the hardware and increase the processing time, as well as power consumption. Power consumption is directly related to the number of hosted servers and running workloads. The peak power consumption became an increasingly important hardware feature in many facilities, since it cannot exceed the power budget provided by the electricity suppliers. This implies that the number of sensors and processing boards that can be installed in a facility depends on the available power budget at a given site. At the same time, a low power budget may not allow to use as many sensors as required to cover a specific area. As a result, the 100% area coverage may not be achieved for some deployments.

All the described requirements that used for this analysis are summarized in Table 5.

Table 5: WDOS Jammer Detection Application's Requirements

Requirement	Description
Availability	99%
QoS (End to End Latency)	90th percentile of the decisions need to be under 400ms
Desired Area Coverage	100%

4.2 System Architecture

To estimate the TCO and area coverage of a specific deployment we use the architecture described in Figure 12. The figure shows that the electricity provider delivers a specific power budget to each site. The figure shows five facilities with different power budgets. Let us assume that three of them (facilities 1, 3, 4) use basic Edge servers for processing the data, whereas facility 2 sends and processes the data in the Cloud and facility 5 uses more power efficient Edge servers to process the data. Figure 12 also shows three bars next to each facility that represent the power budget for each facility (red bar), the cost that depends on the servers that can be operated within the specific power budget (green bar) and the area coverage of the sensors for the specific location which depends on the sensors that can be processed by each server (blue bar). For simplicity, we assume facilities with 100m^2 area. Consequently, to ensure a 100% area coverage for this application, four sensors per facility are required (one per 25 m^2 as described in the application requirements). The figure shows that the more the area coverage is achieved, the more power budget is available, more servers are accommodated and the higher the cost. On the other hand, even though facility 2 provides full area coverage, it needs less power budget because it transfers and processes the data only on the Cloud. However, this option may not meet the QoS time requirement. Finally, facility 5 is shown to have full coverage with the same power budget as facility 4, that only achieves three quarters of the area coverage. This is made possible from the use of more power efficient servers that allow more servers to be operated within the same power budget.

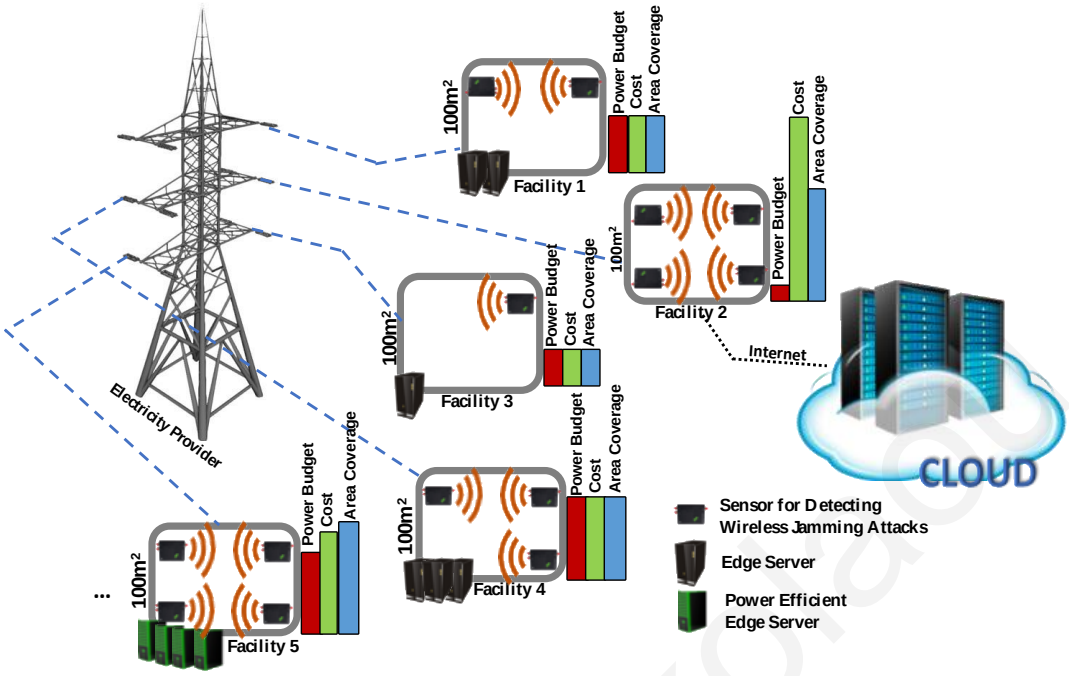

Figure 12: Architecture of Edge servers in different locations

The following equation shows how the area coverage is determined:

$$Area\ Coverage = \frac{EstimatedSensors}{RequiredSensors}$$

This equation shows that area coverage is correlated with the number of sensors (EstimatedSensors) that can be placed in a facility and the required number of sensors (RequiredSensors) that are needed to cover 100% of the specific area which is equal to: (total facility area)/(area covered per sensor). To determine the EstimatedSensors number we use the following equation:

$$EstimatedSensors = MAX_i^{maxInstances} \left\lfloor \frac{PowerBudget}{(ServerPower_i * PUE)} * i \right\rfloor$$

The actual number of sensors that can be deployed is estimated by considering the number of instances per server (number of sensors that data are getting processed on a server). The power of each server configuration that runs specific number of instances is multiplied by the power usage effectiveness (PUE) and then divided by the PowerBudget that corresponds to the power available at a given facility. PUE is a ratio that describes how efficiently a computer DC uses energy. Then, the result is multiplied by the number of instances, i. The

total number of servers that are needed to host the specific number of instances is then obtained from:

$$N = \frac{EstimatedSensors}{i}$$

The number of servers, **N**, and power per server is fed to the TCO model to determine the TCO of the deployment.

The metric that we optimize in this work is the TCO over Area coverage which captures both metrics of interest, as follows:

$$\text{OptimizationMetric} = \frac{TCO}{Area\ Coverage}$$

4.3 Characterization Framework

To improve energy efficiency of a micro-server, we need to investigate the operation limits of voltage and memory refresh rates. Exposing the safe voltage margins of an application is a time-consuming and difficult process due to several abnormal behaviors that can exist [163][164][165][166]. To this end, we developed an automated characterization framework, which is outlined in Figure 13, (1) to identify the target system's limits when it operates at scaled voltage, frequency conditions and DRAM refresh rates, and (2) to record/log the effects of a program's execution under these conditions. The automated framework (outlined in Figure 13) is easily configurable by the user and can be embedded to any Linux-based system, with similar voltage and frequency regulation capabilities. The characterization framework [163][164] consists of three phases (Initialization, Execution, Parsing). During the initialization phase, a user can declare a benchmark list with corresponding input datasets to run in any desirable characterization setup. The characterization setup includes the voltage and frequency (V/F) values on which the experiment will take place and the cores where the benchmark will be run. To reduce the DRAM power, we adopt the framework to characterize DRAM reliability operating under different refresh rates and the supply voltage. Particularly, we use this framework to identify the optimal DRAM refresh rate and voltage which does not trigger uncorrectable errors or system crashes. The execution phase consists of multiple runs of the same benchmark, each one representing the execution of the benchmark in a pre-defined characterization setup. The set of all the characterization runs running the same

benchmark with different setups represents a campaign. In the parsing phase of our framework, all log files that are stored during the execution phase are parsed in order to provide a fine-grained classification of the effects observed for each characterization run.

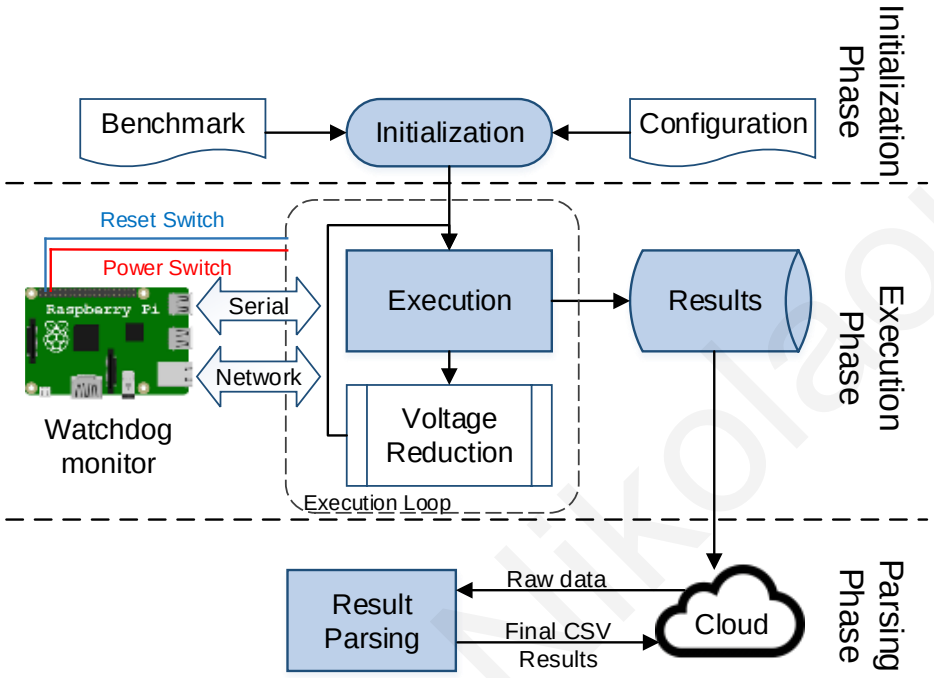

Figure 13: Characterization framework layout

We have also extended the error reporting capabilities of existing mechanisms (i.e. ECC in caches and DRAMs) with system configuration values, sensor readings and performance counters for identifying correctable (CE) and uncorrectable errors (UE). In addition, to account for any undetected error and essentially detect any SDC that could go undetected by ECC, we compare the output of each execution with a golden reference.

The framework provides the following features; it:

- compares the outcome of the program with the correct output of the program when the system operates in nominal conditions to record SDCs,
- monitors the exposed corrected and uncorrected errors from the hardware platform's error reporting mechanisms
- recognizes when the system is unresponsive to restore it automatically,
- monitors system failures (crash reports, kernel hangs, etc.),
- determines the safe, unsafe and non-operating voltage regions for each application for all frequencies, and
- performs massive repeated executions of the same configuration.

4.4 Experimental Setup

4.4.1 Cloud and Edge Architecture

To evaluate Cloud and Edge deployments we use real network traces from both Amazon servers and local servers, respectively. Specifically, for the Cloud evaluation we profile network latency by pinging for one week an Amazon server located in London [167]. London is chosen because, after profiling several sites, it has the lowest network latency. So, it would be obviously a better choice in the Cloud setup. On the other hand, for the Edge evaluation, we profiled for one week the network latency of a server hosted in the same building.

For the compute time, in this analysis we measure the time spend in the ARM processors by emulating the processing board of WDoS application, as shown in Figure 11, because this is the most critical processing component of the application. For the Cloud versus Edge evaluation we considered that the same type of processing board is used either in the Cloud or in the Edge, in order to make them comparable. To estimate the total QoS, we convolute the distributions of network latency results for one week and the compute time results of the processing board.

Several sensors can be attached to each of the processing boards. For this analysis we assume that up to 8 sensors can be attached, equal to the number of cores in the Processing Board's CPU. Each sensor corresponds to one application instance. So, the processing board can collocate a maximum of eight instances. Attaching more sensors per server was not feasible due to the excessively high compute time that leads to QoS violations.

4.4.2 TCO Input Parameters

The main TCO input parameters used in this analysis are shown in Table 6. We consider 8000 sensors, in total, that cover an area of 200,000 m^2. This is representative of a large public building. In the Edge deployment we assume that the micro-servers are distributed in 100 different locations within the building, with 80 sensors located nearby location. On the other hand, for the Cloud deployment all servers are assumed to be placed in one location. For the centralized Cloud, cooling cost, cost of electricity, network cost and the maintenance personnel salary per rack are higher than the Edge configuration. The cooling cost of the Cloud deployment is expected to be much higher than the cooling cost of the Edge deployment as a centralized large datacenter requires much more sophisticated and expensive cooling infrastructure. Regarding electricity costs, the more devices that the centralized

Cloud hosts, the higher power budget is needed and the higher costs are paid to the electricity provider, as this large absolute peak power is reflected in the cost of the electricity. Also, network per rack cost is related to the servers that are placed per rack. For Cloud configuration we assumed racks of 42 servers, whereas for Edge deployments we assumed at most racks with 10 servers, due to the power constrained Edge facilities. In addition, Cloud Power Usage Effectiveness (PUE), is significantly more that the Edge configuration, since cooling is a power-hungry system and uses a non-negligible fraction of the datacenters power. Finally, for Mean Time to Repair (MTTR) of a faulty component we assume that an Edge faulty component can be replaced within 24 hours whereas in a Cloud configuration, the faulty component can be replaced within 1 hour. This difference is primarily justified by considering the geographical distribution of Edge facilities. The MTTR difference is also reflected in the maintenance personnel salary per rack.

Table 6: Edge and Cloud Configurations

	Cloud Configuration	**Edge Configuration**
Total Number of Sensors	8000	
Number of Locations	1	100
Cost of Cooling	3.5 $/kwh	0.019 $/kwh
Cost of electricity	0.08$/kwh	0.07671$/kwh
Network per rack [63]	5000$	1190$
Maintenance salary per rack	208$	8.68$
Power Usage Effectiveness (PUE) [162]	1.3	1.1
Mean time to replace a faulty component	1 h	24 h

4.4.3 Micro-Server Architecture

The server that we use to characterize the WDoS application on, is a state-of-the-art 64-bit ARM based Server-on-Chip, is Applied Micro's (now Ampere Computing) X-Gene 2. X-Gene 2 platform provides knobs for under-volting the various components that are explored. The micro-server consists of eight 64-bit ARMv8-compliant cores running at 2.4 GHz,

grouped in 4 Processor Modules (PMD), which have a separate 32 KB LI instruction cache and 32 KB L1 data cache for each core and a 256 KB unified L2 cache for each PMD. There is also an 8 MB L3 cache which is shared across the whole chip (all 8 cores). There are 4 available memory channels with DDR3 memories.

The X-Gene 2 provides access to a separate Scalable Lightweight Intelligent Management Processor (SLIMpro), a special management core, which is used to boot the system and provide access to on-board monitors for measuring the temperature and power of the SOC and DRAM. The SLIMpro, also reports to the Linux kernel all errors corrected or detected by the provided error-correcting codes (ECC) and the parity. Finally, SLIMpro has configuration parameters of the Memory Controller Units (MCUs), such as refresh period (TREFP). The server runs a fully-fledged OS based on CentOS 7 with the default Linux kernel 4.3.0 for ARMv8 and supports 4KB and 64KB pages.

After characterizing it we choose the voltage levels that do not affect the availability of the system, called safe margins. These margins are used to evaluate the efficient Edge deployment.

4.5 Characterization Results

The WDoS application running on the Processing Board and its dependencies have been ported and tested on the X-Gene 2 host platform by using the characterization framework. The generated results need to be deterministic and repeatable. This is mandatory because in order to detect SDCs among different runs, the output needs to be compared and verified. For this purpose, data sets obtained from recording real life jammer signals, were used as inputs for the application tests. We run the application with various numbers of instances to obtain the trends of the effectiveness. The characterization process reveals the lowest operating limits that achieve the highest power savings without compromising the availability of the system as shown in Table 7.

Table 7: Nominal and Efficient Operating Settings

	Nominal Settings	Efficient Settings
PMD Voltage	980	920
SoC Voltage	950	870
DRAM Voltage	1500	1428
DRAM Refresh Rate in ms	78	2783

Table 8, shows the characterization results running with the nominal and the most energy efficient settings.

Table 8: Characterization results running WDoS application with Normal Setting and Efficient Settings

Running Dos Application with Normal Settings						Running Dos Application with Efficient Settings					
	Idle	1 Instance	2 Instances	4 Instances	8 Instances		Idle	1 Instance	2 Instances	4 Instances	8 Instances
Peak Server Power		67.91	72.81	80.66	88.26	Peak Server Power		63.53	67.64	74.11	80.86
Avg. Server Power	57.43	64.70	69.18	75.79	80.77	Avg. Server Power	54.19	61.21	64.90	70.77	75.17
Avg. PMD Temperature	43.38	59.96	62.15	65.36	69.01	Avg. PMD Temperature	42.98	57.71	58.77	60.81	64.68
Avg. SoC Temperature	44.61	59.41	60.49	62.18	65.17	Avg. SoC Temperature	44.05	58.65	59.29	60.19	62.97
Avg. DRAM Temperature	47.58	62.45	63.06	64.05	66.82	Avg. DRAM Temperature	47.71	62.51	62.91	63.61	66.17

As can be seen from Table 8, the peak and average power can be decreased by 8 and 5 watts, respectively. This reduction corresponds to around 9% savings of the processor power. Moreover, temperature can be decreased by about 3 degrees Celsius and thus, help reduce the need for cooling and can help lifetime reliability.

The findings of the characterization in this Section are used as inputs for the TCO analysis.

4.6 TCO Analysis

This Section reports the TCO analysis of the Edge compared to the Cloud deployment and examines the benefit from more energy efficient micro-servers in the Edge.

4.6.1 Selection of the Number of Instances in Edge and Cloud Deployments

We first evaluate the end-to-end latency of Cloud and Edge deployments in order to select the appropriate number of instances to run in the processing board and at the same time not violate the QoS constraints of the application.

Figure 14(a) and Figure 14(b) show the cumulative distribution of the Cloud and Edge End-to-End latency for different number of instances, respectively.

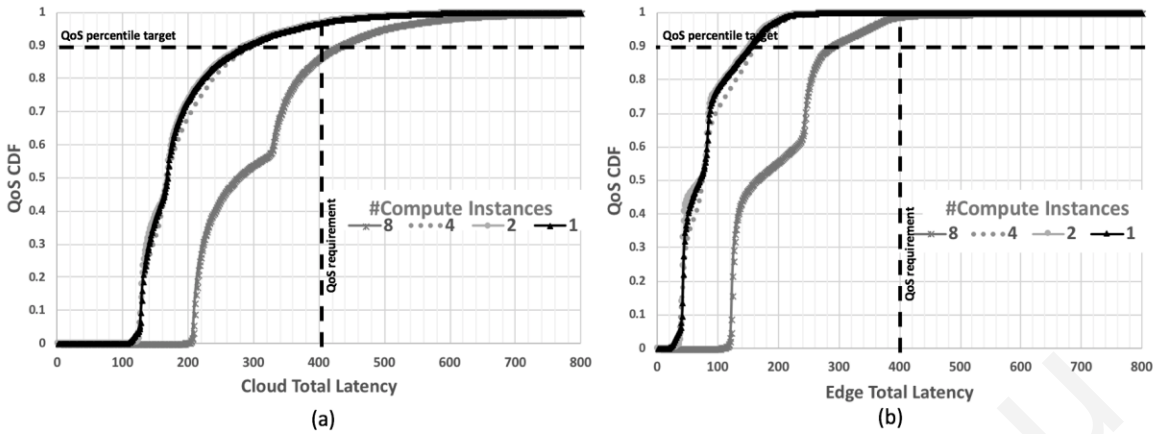

Figure 14: QoS results for different number of instances of the WDoS application running in the Cloud (a) and in the Edge (b)

As the Figure 14(a) shows running 8 instances per server, at the Cloud, is not feasible because this configuration violates the QoS requirement of 400ms for the 90th percentile of the decisions. So, the preferable configuration is the one that uses 4 instances per processing board that corresponds to 4 sensors per board. On the other hand, the QoS results of Edge deployment show that the processing board can simultaneously run 8 instances without violating the QoS requirements of the application. This happens due to the lower network latency of the Edge deployment.

For the rest of the results we use maximum 4 instances per processing board for the Cloud deployment and maximum 8 instances per processing board for the Edge deployment, as well. This requires the use of 2000 servers for the centralized Cloud deployment and 1000 servers for the distributed Edge deployment, in total. In addition, for each Edge locations, there is a placement of 10 servers per location that we assume can operate within the available power at each facility.

4.6.2 Edge Versus Cloud TCO

Figure 15, illustrates the normalized TCO breakdown results with Edge Deployment, for both the Edge and the Cloud. As the Figure shows, the Cloud TCO is 2.13 times higher than the Edge TCO. This corresponds to 80996 dollars more in the Cloud than the Edge facility per month. Particularly, the all the Edge deployments, needed in this analysis spend around 71800 dollars, whereas Cloud spends 152796 dollars per month.

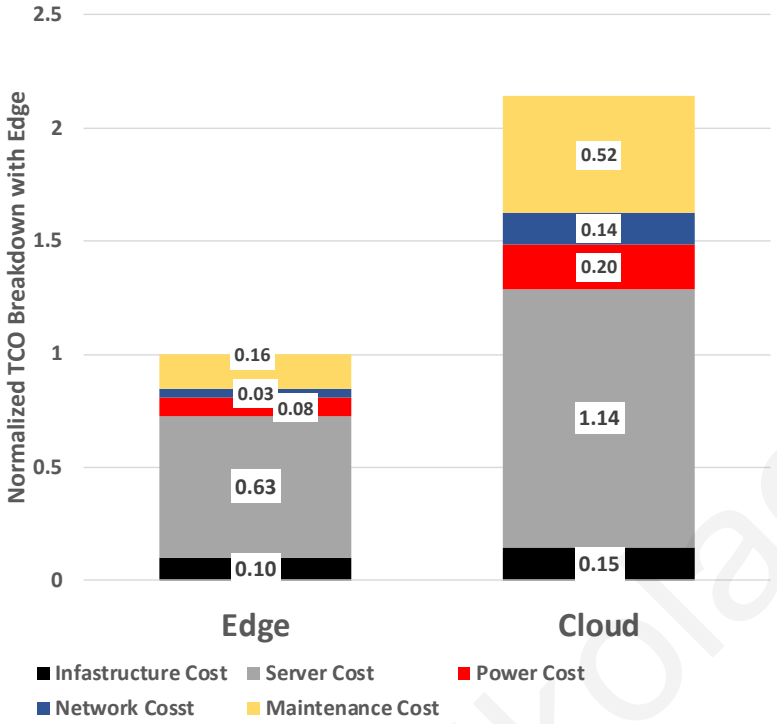

Figure 15: TCO results for Edge and Cloud deployments

This happens, as the breakdown shows, due to the double number of servers that are needed for hosting the total number of sensors to cover the specific area. The obvious difference in the server cost explains the large TCO difference. Also, as Figure 15 shows, the Cloud deployment consumes exactly 2.5 times more power than the Edge deployment. So, except of the double server number, Cloud consumes more power due to the cooling power consumption and the PUE. Additionally, maintenance cost is also around 3.25 times higher in the Cloud than in the Edge deployment. This is due to the lower replacement frequency of the faulty components in the Edge (MTTR). Measuring the availability, we observed that the Cloud can provide four nines of availability (0.9999), whereas, Edge provides only two nines of availability (0.99), which still does not violate the availability requirement of the application.

This analysis highlights that WDoS application can be deployed more efficiently in the Edge than in the Cloud.

4.6.3 TCO and Area Coverage Results for Efficient Edge and Normal Edge Deployments

Figure 16, illustrates an investigation as a function of per Edge site power budgets in Watts (x-axis) and the area in square meters (m^2) that needs to be covered by jamming detector sensors (y-axes).

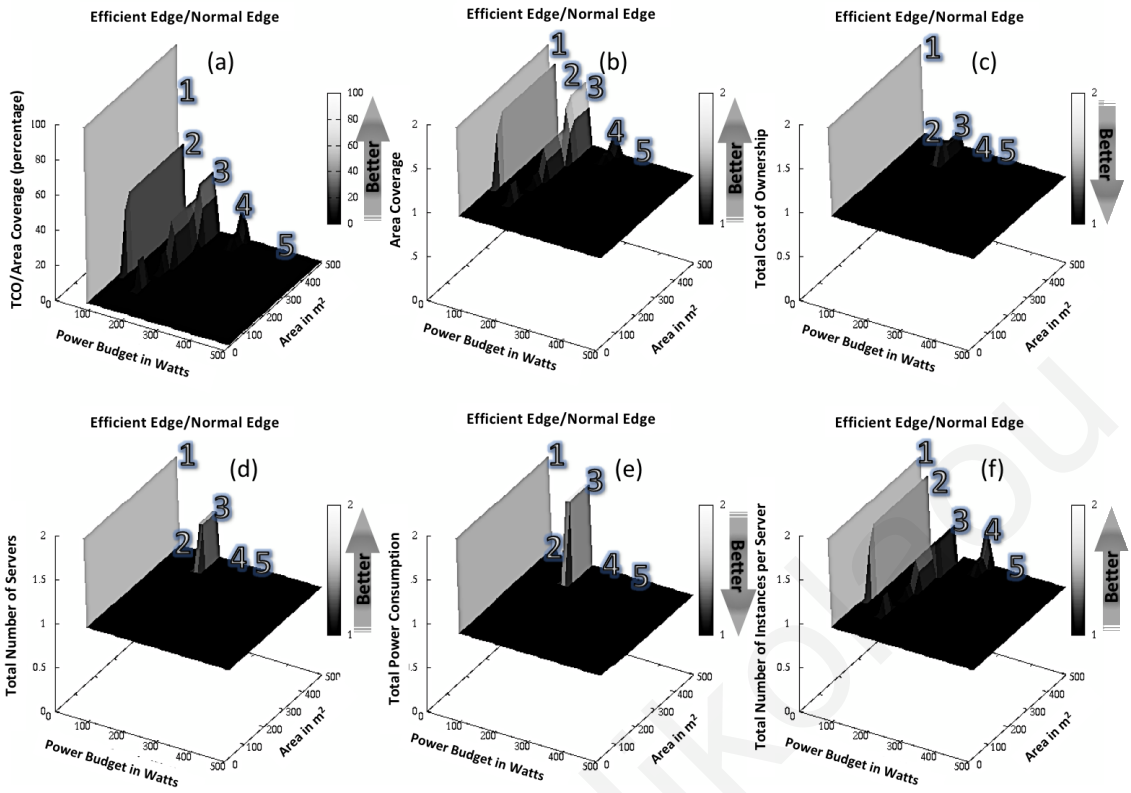

Figure 16: Efficient Edge over Normal Edge results in (a) TCO over Area Coverage, (b) Area Coverage, (c) Total Cost of Ownership, (d) Total Number of Servers that are placed in the deployments, (e) Total Power Consumption, and (f) Total Number of Instances per server

All the presented graphs in Figure 16 show the ratio of Efficient Edge (Edge servers that operate with energy-efficient settings) over Normal Edge (Edge servers that operate with nominal settings). The arrows next to each graph show which direction represents improvement. Also, we use numbered labels for better explanation of the trends in five cases. In all graphs of Figure 16 the labels represent the same case. The first result, in Figure 16(a), shows the metric that we optimize, the TCO over Area coverage. The 3D representation shows that the TCO over the area coverage exposes a sharp increase of the Efficient Edge over the Normal Edge server by reaching the 100% around 73 Watts power budgets for all the cases with label 1.

For this case the normal Edge setup cannot even host one server because the individual server consumes more power than the provided power budget. As the power budget increases, the benefit of the Efficient Edge remains for several power budgets by approaching 60% (label 2), 40% (label 3) and 20% (label 4). After the 400-Watt power budget, the TCO over area coverage has very small difference, around 2% (label 5). The trends in the graph show peaks and valleys due to the discrete power that is needed to fit an extra server, i.e. when power allows the Efficient and Normal Edge to have the same servers the benefits drop (label 5)

and otherwise are high (labels 1, 2, 3, 4). The relative benefits drop as we increase power budget since with higher budget more serves are used and the relative impact of an extra server decreases. But power is a tight resource on the Edge. These benefits shown in Figure 16(a) will be more pronounced with more efficient power servers. The trends observed in Figure 16(a) can be better understood by examining the rest of the graphs in Figure 16. Figure 16(b) shows the area coverage and clearly shows that Efficient Edge can provide always better area coverage and, in some cases, considerably more. These cases correspond to the peak values of Figure 16(a) (labels 1, 2, 3, 4). Also, Figure 16(c), shows the TCO of the (energy-)Efficient Edge over the Normal Edge. This graph shows some points that the Efficient Edge has higher TCO than Normal Edge (for example labels 1, 3). This can be explained by observing the corresponding labels in Figure 16(d) that present the number of servers that can be placed in the deployment. As seen there is a peak on the Figure 16(d) showing that the Efficient Edge uses double number of servers (labels 1, 3). This costs in TCO but provides better area coverage (Figure 16(b), labels 1, 3). On the other hand, these extra servers increase the total power consumption of the deployment, as shown in Figure 8(e) (labels 1, 3). Except these peak numbers in increased power consumption the rest situations of Efficient Edge provide around 9% power savings as compared to the Normal Edge (label 5). The last Figure 16(f), shows the total number of instances placed per server. The number of instances that the Efficient Edge can place in a server is significantly more than the Normal Edge, especially in the peak points of Figure 16(a) (labels 1, 2, 3, 4). Note that the number of servers and the number of instances trends directly correlate with the area coverage trend, in Figure 16(c). This analysis underlines the significance from operating more energy-efficiently on the edge and provides motivation for exploring additional means to increase efficiency on the Edge.

The experimental results, presented in this work, clearly indicate the importance of having power efficiency in the Edge. This observation comes from the analysis of the TCO over area coverage optimization metric that is limited from the power budgeted Edge facility.

As far as we know, this is the first time that all these parameters are explored and analyzed together for evaluation of an IoT application, for both Edge and Cloud deployments. TCO/Area coverage is a useful metric for IoT evaluation and we strongly advocate its use for future IoT application evaluations.

Chapter 5

5 Identification of an Entire Workload's CPU-Vmin and investigation of the Trade-Offs Between Reliability Implications and Power in the TCO.

The end of Dennard Scaling has elevated power into a prime constraint for the CPU design across all market segments. The nominal operating voltage supply of a modern CPU includes worst-case voltage margins [14][23] that ensures correct functionality in the presence of corner-case dynamic and static variations but this limits power efficiency. One method that aims to provide relief from stringent power constraints is under-volting: operate a CPU at a lower than nominal voltage [164][169][170].

A naive approach where a CPU is always undervolted, makes the CPU more susceptible to variations, such as voltage fluctuations or voltage emergencies, which can cause timing violations or bit flips [14] [15] [16] which in turn may lead to silent data corruption (SDC) errors, or even application or system crashes [17][18]. More practical uses of CPU under-volting rely on the characterization and monitoring of a specific CPU chip to identify when to under-volt it and by how-much. A number of works perform an off-line characterization to find the lowest safe voltage (CPU-Vmin) that a CPU can operate correctly for any workload [164][172]. Normal benchmarks or viruses [171], are characterized to identify a common safe CPU-Vmin for a given scenario (e.g. number and location of cores used). Thereafter, in the field when a characterized execution scenario is used, the CPU-Vmin determined during characterization is employed irrespective of the workload executed. This software-based approach relies on a detailed and lengthy characterization process [164][171] and it is vulnerable to failures when a non-characterized workload with higher CPU-Vmin gets executed.

A more application-aware under-volting approach is to monitor on-the-fly an application's microarchitectural behavior and predict whether a voltage emergency is imminent and adjust the CPU supply voltage accordingly [23][173][22][174]. A system in the field normally operates under-volted while monitoring events identified with off-line analysis, when a voltage emergency is observed, it is recorded in terms of a signature that encodes the values

of the different monitored events. The saved signatures are used subsequently to prevent similar problems in the future. This approach has more potential than the application-agnostic under-volting tactic, but in previous work it is mainly evaluated with simulation and requires hardware support for fine-grain, cycle scale, event monitor/recording, as well as, checkpoint/rollback or restart techniques to recover from emergencies [22][23][174][176][19].

Another undervolting approach uses an off-line characterization to correlate correctable cache errors and supply voltage to drive an online under-volting governor that decides the supply voltage based on the correctable cache error-rate [170]. This approach is not applicable to platforms where SDC or detectable uncorrectable errors (DUE) occur before correctable cache errors are observed [164].

This Chapter presents, for the first time, as far as we know, that with the monitoring of selected performance counters, determined using an off-line analysis of other workloads, a workload can reveal its safe CPU-Vmin during the first-n seconds of its execution. Upon a workload's CPU-Vmin identification, the execution can continue at the selected CPU-Vmin for the rest of its execution without experiencing any instability or failures. This workload aware approach is software-based and does not require any hardware support

5.1 Background

A variety of prior works has been focused on the characterization of CPU-Vmin for different workloads [164] [172] [184]. These works provide characterizations for pessimistic voltage guard bands for core-to-core and chip-to-chip variations for online workload-agnostic executions. All these works use an offline characterization and an online workload-agnostic, CPU-Vmin prediction based on core allocation. Moreover, George et al. [184] shows that multicore and multithreaded workloads operate at similar CPU-Vmin.

On the other hand, there is a number of works that are workload aware. All these works use microarchitectural events to predict voltage emergencies during the execution of a workload [22][23][173][174]. Gupta et al. [173] shows a correlation of voltage noise with some of the architectural events such as cache misses, TLB misses and long-latency stalls and proposes compiler-based optimizations to reduce these events and thus to prevent from voltage emergencies. Another work of Gupta et al. [174] proposes to avoid voltage emergencies by adding some pseudo-nops and prefetching. Reddi et al. in [23] and [22] use microarchitectural events such as pipeline stalls and L2 misses to predict voltage emergencies for single core CPUs. However, all these works are based on simulations and

not on real hardware. Thus, the granularity of the voltage emergency detection is in terms of cycles. This work shows that by taking specific counters into account, we can identify the workloads CPU-Vmin in the first n-seconds of a workload's execution. As far as we know, there is no other work that shows that the first n-seconds can identify the CPU-Vmin for the rest of a workload's execution.

5.2 Experimental Setup

To achieve the identification of a workload's CPU-Vmin, we firstly provide characterization of different multi-program and multi-threaded workloads using real hardware. The characterization classifies workload into two categories, V-low and V-high. V-low workloads are the workloads that can operate at the lowest CPU-Vmin observed during the characterization, while the V-high are the workloads that require a CPU-Vmin higher than the V-low to operate correctly.

5.2.1 Platform

This study is performed using an X-Gene2 server. The X-Gene2 server's CPU consists of eight 64-bit ARMv8-compliant cores running at 2.4 GHz, grouped in 4 Processor Modules (PMD) [178]. Each PMD contains two high-performance X-Gene2 cores, each of which has its own 32 KB L1 I-cache, 32 KB L1 D-cache, and Floating-Point Unit (FPU). The pair of X-Gene2 cores in a PMD shares a 256 KB L2 cache unit which interfaces to Central Switch (CSW) interconnect. All four PMDs share an L3 cache (8 MB), which is also attached to the CSW.

The X-Gene2 provides access to a separate Scalable Lightweight Intelligent Management Processor (SLIMpro), a special management core, which is used to boot the system and provide access to on-board monitors for measuring the temperature and power of the SoC and DRAM. The server runs a fully-fledged OS based on CentOS 7 with the default Linux kernel 4.3.0 for ARMv8.

X-Gene2 consists of three independent voltage domains, the PMD, SoC and DRAM domains and provides knobs for under-volting each of the three domains, independently. PMD domain contains the cores, the L1 instruction and data caches and the L2 cache. The SoC domain contains the L3 cache, the DRAM controllers, the central switch and the I/O bridge. Finally, the DRAM domain contains all the DIMMS.

In this work we study only under-volting operation of the PMD domain that contains the eight cores of the CPU. The nominal voltage setting for the PMD is at 980mV. The nominal settings for the other domains are 950mV for the SoC and 1500mV for the DRAM, which for this analysis remain constant.

5.2.2 Workloads

The benchmarks used for this analysis are collected from 4 different benchmark suites. Particularly, we use 17 benchmarks from SPEC CPU2006 Suite [177], 14 benchmarks from SPEC CPU2017 Suite [179], 6 benchmarks from PARSEC Parallel Benchmark Suite v3.0 [180] and 8 benchmarks from NAS Parallel Benchmark Suite v3.31 (NBP) [54]. SPEC CPU2006 and SPEC CPU2017 are single-thread benchmarks, while NAS and PARSEC are multi-thread benchmarks. We executed all the 45 benchmarks using always all 8 cores, fully utilized. Thus, for the single-thread benchmarks we run 8 instances of the same benchmark in the X-Gene2. In this case each instance is pinged in a single core. For the multi-thread benchmarks we set them to run using 8 threads on the X-Gene2 machine.

5.2.3 Vmin Characterization

To investigate the CPU-Vmin of the PMD domain of X-Gene2 we followed an automated characterization process. The characterization process [164][181] consists of three phases (Initialization, Execution, Parsing). During the initialization phase, the benchmark list is determined, as described in Section 5.2.2.

The execution phase consists of multiple test of the same benchmark at incrementally lower voltage. Particularly, we perform each benchmark test 10 times where each test includes multiple runs of a benchmark, each with lower voltage at steps of 10mV.

Finally, during the parsing phase, we determine the CPU-Vmin voltage: the minimum voltage observed without any crash in all the tests of a specific benchmark. Note that we also check for any undetected error essentially for silent data corruption (SDC) errors, by comparing the output of each execution with a golden reference which is derived from running at nominal settings. The safe CPU-Vmin for each benchmark, is the minimum voltage that does not experience any crash, SDC errors or other unexpected behavior in all the 10 tests.

Table 9: Benchmarks with their CPU-Vmin and Execution Time

#	Benchmarks	CPU-Vmin (mV)	Ex. Time (sec.) – 20s/ex. time(%)	#	Benchmarks	CPU-Vmin (mV)	Ex. Time (sec.) – 20s/ex. time(%)
V-low							
1	SPEC2006_zeusmp	900	1203 - 1.66%	**13**	SPEC2017_parest	900	1002 - 2%
2	SPEC2006_povray	900	445 - 4.49%	**14**	SPEC2017_ometpp	900	1857 - 1.08%
3	SPEC2006_ometpp	900	771 - 2.59%	**15**	SPEC2017_leela	900	775 - 2.58%
4	SPEC2006_hmmer	900	268 - 7.46%	**16**	SPEC2017_fotoik3d	900	1497 - 1.34%
5	SPEC2006_h264ref	900	154 - 12.99%	**17**	SPEC2017_exchage2	900	1910 - 1.05%
6	SPEC2006_gromacs	900	1538 - 1.30%	**18**	SPEC2017_deepsjeg	900	708 - 2.82%
7	SPEC2006_GemsFDTD	900	1435 - 1.39%	**19**	Parsec_fluidaimate	900	58 - 34.48%
8	SPEC2006_gamess	900	140 - 14.29%	**20**	NPB_mg.C.x	900	45 - 44.44%
9	SPEC2006_bzip2	900	1073 - 1.86%	**21**	NPB_lu.C.x	900	264 - 7.58%
10	SPEC2017_xalacbmk	900	708 - 2.82%	**22**	NPB_is.C.x	900	6 - 100%
11	SPEC2017_roms	900	525 - 3.81%	**23**	NPB_cg.C.x	900	121 - 16.53%
12	SPEC2017_perlbech	900	945 - 2.12%	**24**	NPB_bt.C.x	900	311 - 6.43%
V-high							
25	SPEC2006_milc	910	1019 - 1.96%	**36**	SPEC2006_amd	920	815 - 2.45%
26	SPEC2006_mcf	910	1329 - 1.50%	**37**	Parsec_streamcluster	920	78 - 25.64%
27	SPEC2006_bwaves	910	1772 - 1.13%	**38**	Parsec_freqmie	920	41 - 48.78%
28	SPEC2006_soplex	910	931 - 2.15%	**39**	Parsec_facesim	920	81 - 24.69%
29	SPEC2017_povray	910	1176 - 1.70%	**40**	Parsec_bodytrack	920	38 - 52.63%
30	SPEC2017_lbm	910	961 - 2.08%	**41**	NPB_ua.C.x	920	285 - 7.02%
31	SPEC2017_imagick	910	587 - 3.41%	**42**	NPB_sp.C.x	920	334 - 5.99%
32	SPEC2017_cactuBSSN	910	1524 - 1.31%	**43**	SPEC2006_toto	930	1359 - 1.47%
33	Parsec_blackscholes	910	32 - 62.50%	**44**	SPEC2017_cam4_r	930	376 - 5.32%
34	SPEC2006_wrf	920	1679 - 1.19%	**45**	NPB_dc.B.x	930	396 - 5.05%
35	SPEC2006_gobmk	920	882 - 2.27%				

Table 9 shows the CPU-Vmin characterization results for all the 45 benchmarks on the X-Gene2 platform. The results are grouped into four categories according to the observed CPU-Vmin of each benchmark. As shown in Table 9, 24 benchmarks can operate at 900mV, 9 benchmarks at 910mV, 9 benchmarks at 920mV and 3 benchmarks at 930mV.

From the data we observe that both single-threaded and multi-threaded benchmarks appear in all the four categories and all benchmarks can operate at least 50mV lower than the nominal voltage. Another key observation is that, a significant number of benchmarks operates at 900mV (more than half of the benchmarks, 24/45). For simplicity reasons in this study we classify the benchmarks only into two categories, the V-low (top-half) and V-high (bottom-half) categories as can be seen in Table 9.

V-low category consists of the benchmarks that operate at 900mV and V-high category consists of the benchmarks that operate at all the other voltages which are higher than 900mV. In particular, V-high includes all the benchmarks that operate at 910mV, 920mV and 930mV. Our study can be extended to consider more than two categories, but we leave that for future work.

Table 9 also lists the execution time of each benchmark. The execution time of the single-thread benchmarks is on average around 1000 seconds, whereas for multi-thread benchmarks is on average around 150 seconds.

5.3 Performance Counters Selection Methodology

A large number of performance counters exists in X-Gene2 and it is critical to choose the ones that have the highest correlation with the CPU-Vmin. We first explore the 21 performance counters of X-Gene2 listed in Table 10. We collected statistics for all the performance counters at one-second granularity. We then labelled each benchmark's performance counter measurements with its CPU-Vmin and performed a Pearson-Correlation for each performance counter and the CPU-Vmin across all benchmarks. This produces a correlation coefficient per performance-counter between -1 to1. Performance-counters that have a correlation-coefficient above a threshold (absolute value 0.7) are selected. Guided by

Table 10: Performance Counters in X-Gene2

	Performance Counters
1	Utilization
2	Instructions
3	Cycles
4	L2 accesses
5	L2 misses
6	L3 misses
7	branch misses
8	Syscalls
9	L2 prefetch
10	Exception taken
11	L1 misses
12	L1 TLB misses
13	Decode starved
14	op dispatch stalled cycle
15	BTB misprediction
16	Branch speculative executed - Indirect branch
17	Branch speculative executed - Immediate branch
18	L1 data TLB refill - Write
19	Operation speculatively executed - Integer data processing
20	Operation speculatively executed - Advanced SIMD
21	Operation speculatively executed - FP

these results we selected the following five performance counters as the most useful for CPU-Vmin correlation: syscalls, L2 prefetching, exception taken, L1 ITLB misses and BTB mispredictions (colored with blue in Table 10). The combination of these performance

counters is selected because it exhibited the highest correlation with increasing CPU-Vmin as compared to any other combination based on a subset of these counters. More specifically, Figure 17, shows the Pearson-Correlation coefficient between CPU-Vmin and each of the 31 possible combinations that use one or more of the five performance counters. The figure clearly shows the synergy from combining more counters together and that the combination that uses all five performance counters provides the highest correlation with CPU-Vmin. Moreover, we also observed (not shown in the results) that adding any one of the other performance counters, listed in Table 10, does not lead to a higher correlation coefficient. Thus, we need to consider all the five counters in combination. Additionally, we noticed that four of the counters - syscalls, exceptions taken, L1 TLB misses and BTB mispredictions - exhibit a positive correlation, whereas L2 prefetching exhibits a negative correlation.

This means that when the number of syscalls, exceptions taken, L1 TLB misses and BTB mispredictions is high but the number of L2 prefetches is low, then the voltage is virtually always V-high as well

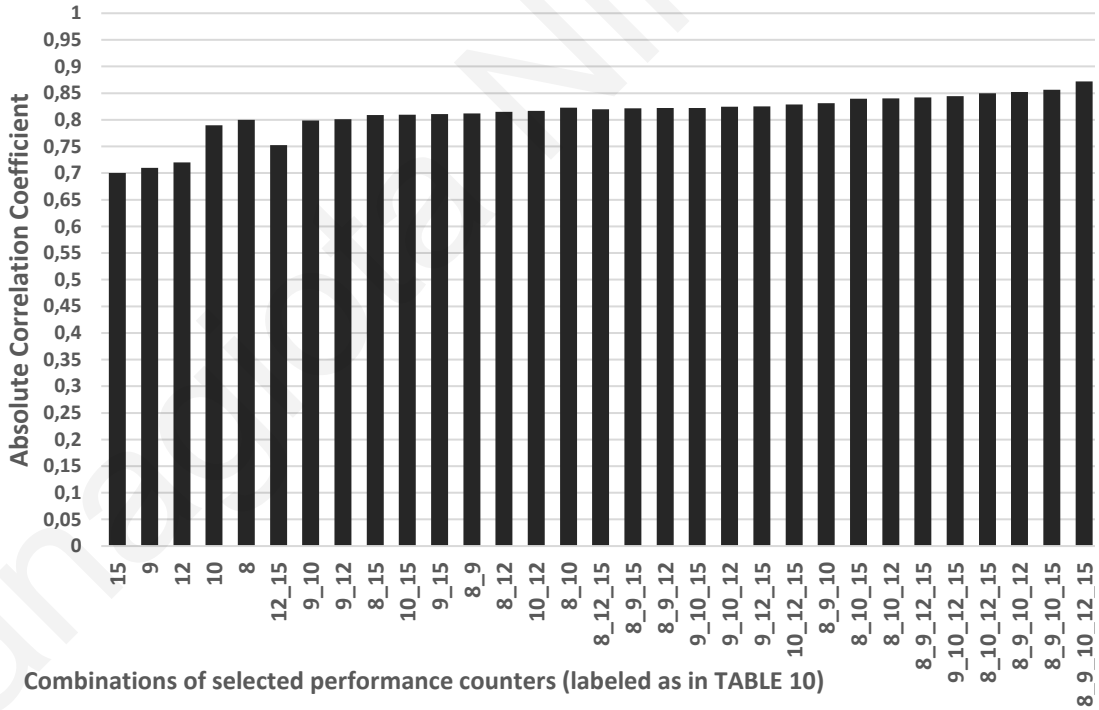

Figure 17: Correlation Analysis between different Combinations of the Selected

Related work [23][173][22][174] has concluded that a combination of microarchitectural events, corresponding to the abovementioned counters, in a short time span (few cycles) often leads to a voltage emergency. However, these earlier works do not show any correlation with L2 prefetching. We hypothesize that this is due to the use of simulation for their evaluation and not real hardware measurements, as we do in this work. In particular, we believe that the correlation to low L2 prefetching captures low L1 cache misses, high IPC

and high-power consumption that causes a drop in supply voltage. It is during such time when all the other events (syscalls, exception taken, L1 TLB misses and BTB mispredictions) occur that cause a voltage droop that is combined with the voltage drop and leads to a voltage emergency to happen. Voltage-droops are caused by a sudden change in the activity of the circuits powered by a voltage domain [175]. In summary, the correlation analysis suggests that a method that aims to classify the CPU-Vmin of a workload into V-low or V-high, for the platform we are using (X-Gene2), it needs to consider together the combination of all five selected performance counters. This combination of the five selected counters is termed as signature to the rest of this Chapter.

5.4 Workload's CPU-Vmin Identification Method

Figure 18 illustrates with the help of an example the high-level functionality and objective of the proposed CPU-Vmin identification method. Particularly, this methodology aims to detect, after the first n-seconds of a workload's execution, its CPU-Vmin. Figure 18.a shows the behavior with a workload predicted to have V-low CPU-Vmin. When its execution starts, the voltage is set to a safe setting (V-high) and the signatures (selected five performance counters) are collected per second until the execution time reaches a threshold. At that point, the workload's signatures are compared with the signatures derived from an offline characterization/training of other V-low workloads. When each of the collected signatures, of the currently running workload, matches with a signature in the training set, the workload is identified as V-low and thus the voltage is reduced to 900mV. On the other hand, a workload that produces, before its execution reaches the time threshold, a signature not found in the training set of signatures, is identified as V-high (Figure 18.b)), and the execution continues at 940mV. The choice of 940mV as safe voltage, is based on the highest voltage that we observe (930mV,Table 9) for V-high benchmarks that gets increased by a 10mV margin to ensure the safe operation for all the benchmarks that are used in this work. An alternative, for a higher safety or due to limited analysis of V-high workloads, is to use the nominal voltage for the first (identification) phase of the execution of all workloads as well as for the remaining execution for the workloads classified as V-high. Operation with nominal voltage avoids unexpected events, such as SDC or crashes, and, consequently, avoids the risk to need recovery for workloads classified as V-high.

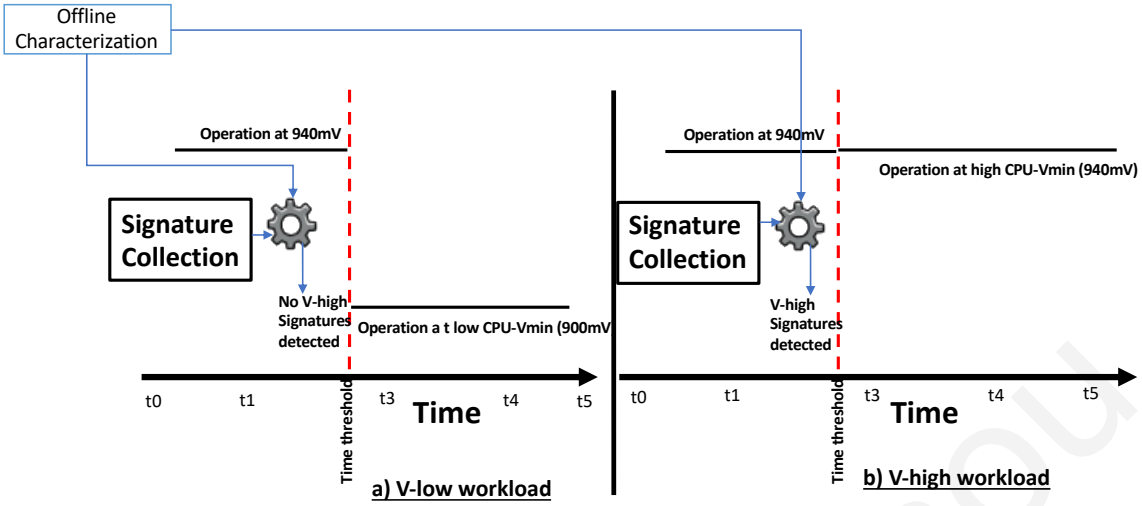

Figure 18: Identification Method Example for a) V-low workload and b) V-high workload

As this is a predictive method, there is a possibility that the classification into a V-low or V-high workload to be incorrect. Specifically, when a V-low workload is identified as V-high or a V-high workload as V-low. Definitely, the second case is the one that needs to be avoided because it can lead to SDC errors and application or system crashes. In contrast, the ramification of a wrong identification of a V-low workload as V-high is lower power savings.

5.4.1 Performance Counter's Signature Semantics

Variability in the values of the monitored performance counters results in generating a large number of unique signatures during training which makes the classification of the CPU-Vmin more expensive and complex. The value variability, also, renders the proposed method ineffective, as small differences between two signatures will result in classifying a V-low benchmark as V-high (false-positive) and, thus, reduce the potential power savings.

We address the issues of value variability and large number of signatures by applying clustering to the monitored values. In particular, instead of using in a signature the original performance counter value, we use its corresponding integer logarithmic value (the logarithm of a value without any decimal digits). The Log function is often used in various clustering or outlier detection problems [182][183]. To this end, we explore logarithms with different bases (log2 through log10) to find the most effective clustering function that filters out redundant information. Our evaluation revealed that log2 is an appropriate clustering function that helps distinguish efficiently V-low from V-high signatures. Figure 19 helps visualize the effectiveness of the proposed clustering function. It shows a two-dimensional projection of

our five-dimensional signatures based on R-tool's fviz_cluster projection [81]. Each data point is a two-dimensional representation of a signature with five values each of which has been transformed with our clustering function. This projection aims to retain in the two-dimensional space the distance between signatures in the five-dimensional space, i.e. points that are close/far in the original multi-dimensional space are also close/far in the two-dimensional space. With blue color, we represent the signatures of V-low workloads and with red color the signatures of V-high workloads. The figure also draws a blue area that encloses all V-low signatures and a red-area that encloses all V-high signatures. As Figure 19 shows, V-high and V-low areas have considerable overlap (we cannot differentiate a V-high signature in this overlapped region). However, at the top-right, the V-high area is separate from the V-low area. This area contains signatures found in V-high workloads that are distinct from V-low signatures. We examined these V-high signatures and found them to have, as compared to signatures in the V-low area, higher number for syscalls, exceptions taken, L1 TLB misses and BTB mispredictions and lower number of L2 prefetches. This is in agreement with the observations in Section 5.3 regarding which performance counters exhibit the strongest correlation with the CPU-Vmin and the sign of their correlation coefficient.

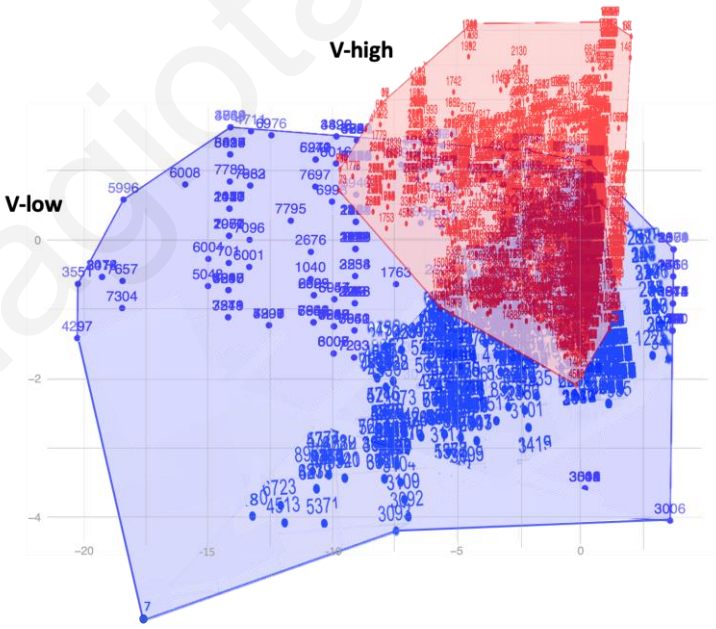

Figure 19: Signature's clusters for V-low and V-high benchmarks

5.4.2 Workload CPU-Vmin Identification Method

In this Section, we present a detail overview of the proposed CPU-Vmin identification method.

Figure 20 shows that the method consists of an offline and an online phase. During the offline phase, we first determine the CPU-Vmin for each workload in a given set of training workloads. This is accomplished using a CPU-Vmin characterization procedure as outlined in Section 5.2.3. Then, for those workloads that have V-low CPU-Vmin, we collect their performance counter signatures per second. Each signature contains the number of syscalls, exceptions taken, L1 TLB misses, BTB mispredictions and L2 prefetches. Once the signatures of all V-low workloads in the training set are collected, we apply to them the clustering function to produce the final set of signatures (refer to as V-low signatures) that will be used during the online phase to predict whether a workload's CPU-Vmin is either V-high or V-low.

During the online phase, an (unknown) workload starts executing using a safe CPU-Vmin and its signatures are collected for the first-n seconds of its execution. After n seconds have elapsed, the collected signatures are transformed with the same clustering function used during the offline phase. The resulting signatures are checked against the V-low signatures (those generated during the offline phase). If each signature, from the first-n seconds of the execution of the unknown workload, matches with a V-low signature then the workload is classified as V-low otherwise, i.e. at least one signature of the unknown workload is not found in the V-low signatures, the workload is classified as V-high. If the classification is correct, the operation is safe for both (V-high and V-low) workloads and power is reduced for the V-low workloads. If the classification is incorrect, operation is safe but power is not saved for V-low workloads that are classified as V-high. Finally, SDC and crashes may occur for workloads that are V-high but classified as V-low. A central parameter of the method, and of course of our evaluation, is the time threshold that is used to identify a workload's CPU-Vmin. Our method is based on the hypothesis that benchmarks reveal early during their execution whether they are V-high or V-low. The longer the time threshold the more signatures are produced and checked and, therefore, the less chance for false-negatives (a V-high workload classified as V-low) but this comes at the expense of lower power savings since the voltage of correctly classified V-low workloads gets lowered later during their execution. Our evaluation investigates the interplay of the time-threshold used for identification and false-negatives.

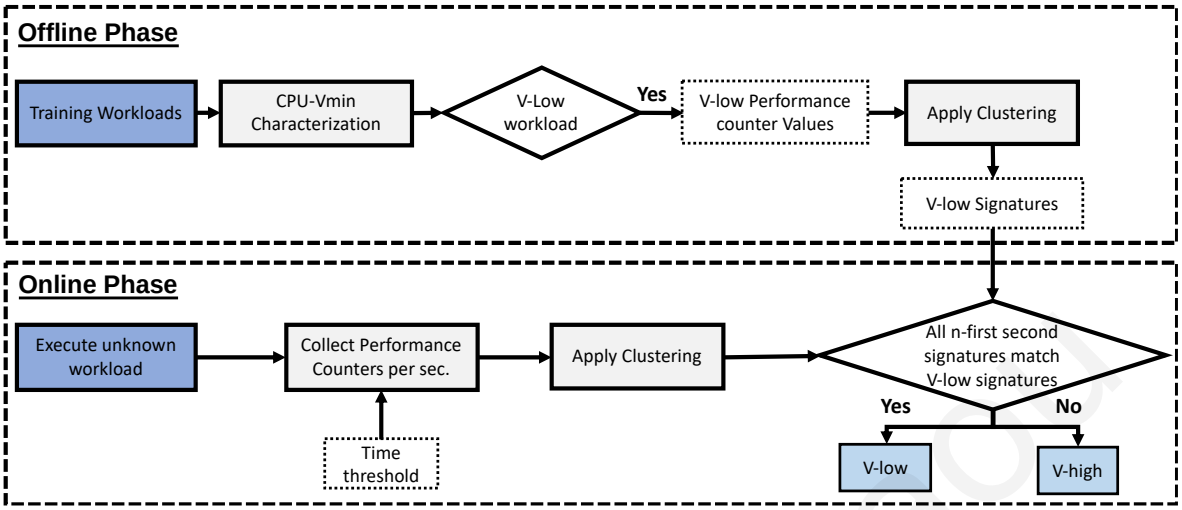

Figure 20: CPU-Vmin identification framework with the input parameters and output decisions

5.5 Experimental Methodology

To evaluate the effectiveness of the CPU-Vmin identification method, we use the X-Gene2 platform and the 45 benchmark set listed in Table 9. SPECCPU2006 and SPECCPU2017 benchmarks include runs with multiple inputs. For tracking the five selected performance counters, we use values obtained from the perf tool every second.

For collecting power measurements, we use power monitors that exist in the X-Gene2 platform. To assess the power savings of the CPU-Vmin identification method we compare the power consumption of our method with the operation that is always at 940mV. The 940mV is derived from the offline characterization of the CPU-Vmin for all the 45 benchmarks. We also show the power savings obtained when compared to operation at nominal voltage settings (980mV).

For a thorough evaluation of the identification method, we followed a cross-validation evaluation approach. In particular, we perform multiple experiments where for each experiment, two workloads, one V-low and one V-high, are removed from the benchmark set and are used as the unknown workloads for testing. The remaining (23) V-low benchmarks are used for training.

We assess the accuracy of the proposed CPU-Vmin identification method by comparing the CPU-Vmin predicted for each workload during the testing phase with the actual CPU-Vmin of each workload. Thus, a true-positive (TP) corresponds to the case where the testing phase classifies a workload as V-low and it is actually V-low. True-negative (TN) represents the case where the testing phase determines a workload as V-high and the workload it is indeed

V-high. A false-positive (FP) occurs when the workload is classified as V-high but the workload is V-low. Although, false positives are undesirable, they are tolerable as they only lead to less power savings without any crashes. Finally, a false-negative (FN) corresponds to the case where a workload is classified as V-low in the testing phase but it is actually V-high. Thus, the voltage is reduced to V-low after the first-n seconds and with high likelihood this will lead to the application or system to crash.

We performed in total 504 different experiments (21 V-high x 24 V-low combinations). As a result of our evaluation methodology, the tested V-low workload can be categorized only as TP or FP and the tested V-high workload only as TN or FN. So, for each of the 504 total experiments a pair of workloads are tested separately, the one is tested for TP/FP and the other for TN/FN.

5.6 Results

5.6.1 Accuracy

We first evaluate the accuracy of the CPU-Vmin identification method. Figure 21 plots the accuracy across the 504 different experiments when the time threshold is set at 20 seconds. The Figure shows the breakdown of TP, FP, TN and FN for all the 504 experiments. The results are summarized as 209/504 TP, 295/504 FP, 501/504 TN and 3/504 FN.

We observe that the method is quite accurate in classifying V-high workloads. Specifically, only for three out of the 504 experiments (0.6%) the method failed to identify correctly a V-high workload. For such cases, a fail-safe technique, such as checkpoint/rollback, is essential for recovery. A crashed workload after recovery can resume execution at a safe voltage (940mV or even at nominal).

It is useful to point out that for benchmarks with multiple inputs the identification method determined the CPU-Vmin only during the first 20s of the execution with the first input (i.e. in our experiments a benchmark with multiple inputs is classified only once). We have also confirmed that the obtained results are insensitive to the order that a benchmark executes its different inputs. It is also important to point out that many of the benchmarks used in this analysis, consist of multiple phases, specifically, 37 out of 45 benchmarks. We determined that a benchmark has multiple phases when at least one performance counter exhibited considerable difference between the signatures collected over a workload's entire execution.

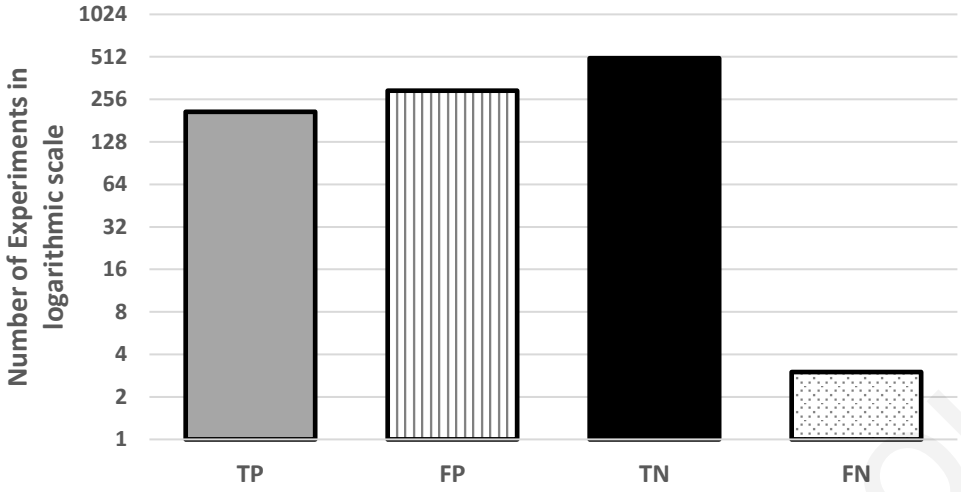

Figure 21: Identification Framework Accuracy results

All the above provide a strong support for the key hypothesis: for the platform and benchmarks used in this study, for a given workload execution, a safe CPU-Vmin can be identified with very high accuracy during the first 20-seconds of its execution. As shown in Table 9, for the benchmarks used in this study, the 20s correspond to at least to 1% of the total execution time and this seems to be, in general, sufficiently long for a workload to reveal whether or not it requires a V-high CPU-Vmin. Of course, it is important for the method to not often classify V-low workloads as V-high and loose power saving opportunity. The results show that the FP rate is 58% (295/504). Therefore, our method saves power 42% of the time but misses on a considerable power reduction potential.

We checked the experiments that suffer a FP and we observed that they happen consistently each time we test a V- low benchmark that comes from a specific subset of 14 benchmarks. Consequently, each of these 14-benchmarks has at least one signature that does not belong to any other (23) V-low benchmark. Thus, our method classifies these benches as V-high. We have also checked if these distinct signatures overlap with the signatures of the V-high workloads. We have found that at least one signature from these benches does not belong to the signatures of the V-high workloads. This distinct behavior of the FP benchmarks can be visualized in Figure 22. The figure uses the same two-dimensional projection of the five-dimensional signatures of all workloads but with different color (green) for the signatures from FP workloads. It is clearly visible that there are signatures of FP benches that are not overlapped with either V-low or V-high workloads. This finding provides a motivation for future work to reduce the FP rate.

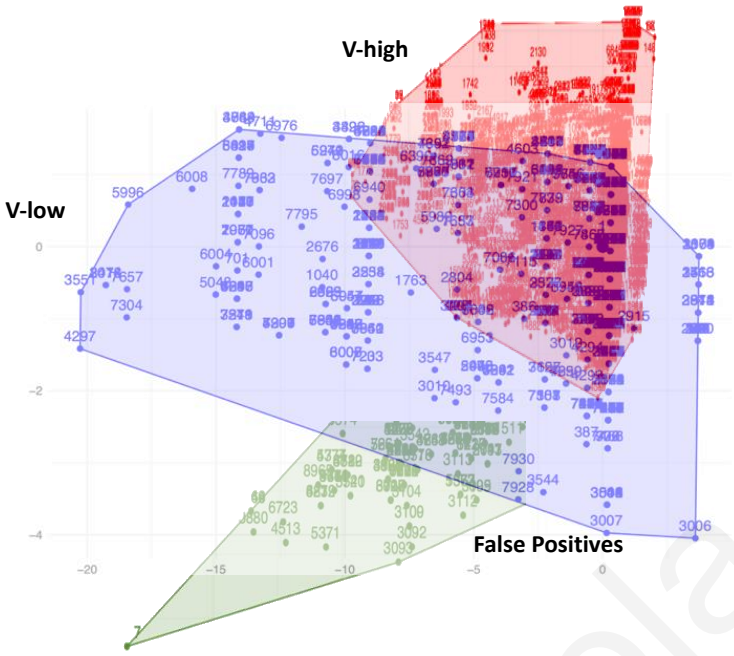

Figure 22: Signature's clusters for V-low, V-high and false positive benchmarks

5.6.2 Time Sensitivity Analysis

One of the main objectives of the proposed method is to identify quickly the CPU-Vmin of a workload. Thus, we perform an analysis of the sensitivity of the method to the time threshold used to classify a workload.

Figure 23, shows the distribution of the identification time for all the 501 experiments that classify correctly a V-high workload (TN). The figure shows only the appearance time of the first V-high signature. As the figure reveals, for more than half of the experiments a V-high workload is identifiable after the first second (268/501). For the other cases, the identification happens few seconds later but no later than 20s. We note that by setting the time-threshold at 10 seconds, the TN accuracy of the method drops from 99.4% to 91%, at 5 seconds to 86% and at 2 seconds to 53% (correspondingly the FN rate increases from 0.6% to 9%, 14% and 47%). We have examined threshold-times beyond the 20 seconds but we observe no change in the TN/FN rate. Consequently, a time-threshold of 20 seconds for CPU-Vmin identification appears the most appropriate for our setup and workloads.

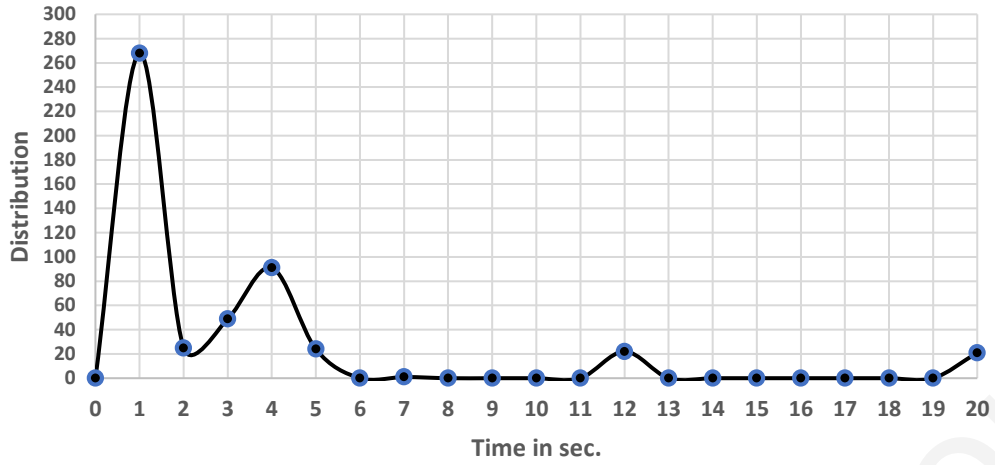

Figure 23: Identification Time Distribution of the first V-high signature appearance in the 501 V-high benchmarks

For the majority of the workloads (27 of 45) the first 20 seconds of execution correspond to less than 5% of their total execution time. The fraction of the execution that 20 seconds correspond to for each workload is listed in Table 9. Although, identifying a workload earlier enables higher power savings we need to keep the time threshold at 20 seconds to avoid increasing FN rate and the occurrence of SDCs and crashes.

5.6.3 Power Evaluation Results

Figure 24 presents the power savings of the proposed method for each of the 504 experiments. The experiments are sorted in increasing order of power savings that is obtained for the V-low workload tested in each experiment. More than half of the experiments have no power reduction since they are FP (V-low workload classified as V-high). For the other experiments (TP), the benefit is close to 8% (15%) as compared to when operating with a supply voltage of 940mV (nominal 980mV). Figure 24 also shows the average power savings which are 3.8% compared to the operation at 940mV and 7.1% compared to the operation at 980mV.

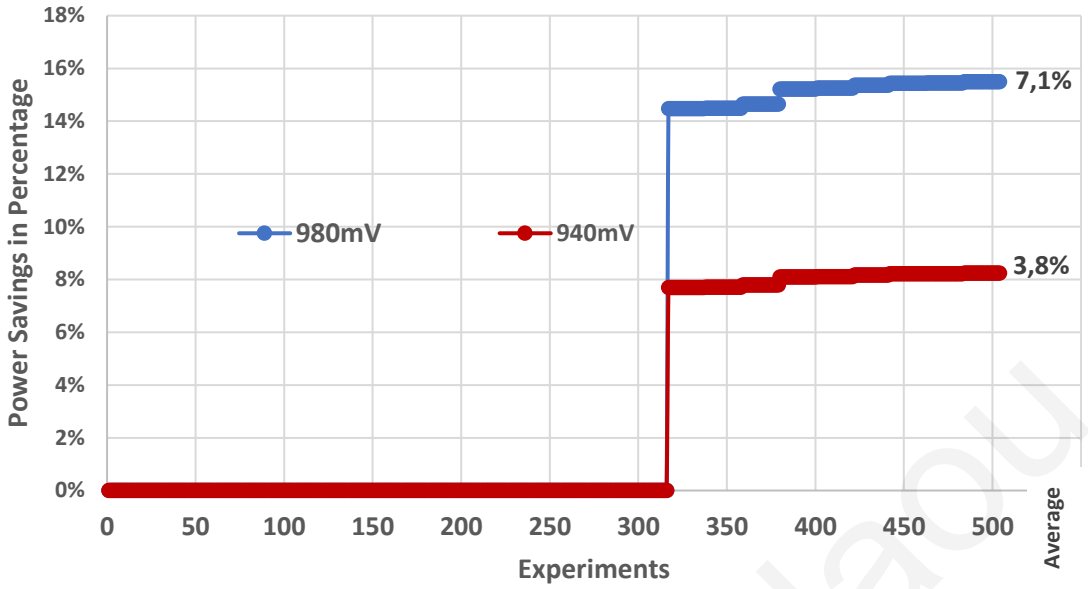

Figure 24: Power savings results for all the experiments compared to the baseline that operates at 940mV and at 980mV.

5.6.4 TCO Evaluation Results

Next, we evaluate the TCO for various detection times shown in Section 5.6.2 (20sec., 10sec., 5sec. and 2 sec.). To mitigate crashes, we assume a checkpoint/restart technique that takes 5 minutes. Also, we assume that all the 504 experiments run in parallel, each in a different server. Thus, our baseline is a DC with initial 504 servers. Figure 25, presents the TCO results for the different detection times (20, 10, 5 and 2 seconds). Also, this Figure depicts the availability values in percentage. The presented TCO is relative to the 20 second's results. From this Figure, we observe that TCO is better when the detection time is at 2 seconds. This happens because, even though the system provides lower availability, the power gains are more than all the other three detection times. However, real systems need to ensure specific levels of availability. The target availability in this case is 99%. Figure 26 depicts the same experiment maintaining availability always above 99%. Furthermore, each detection time bar is labeled with the absolute TCO values in dollars. As the Figure shows, 20 seconds detection time is beneficial in TCO by providing almost double savings from all the other three detection times. This is due to the number of servers that are needed to ensure the required levels of availability and thus, cover for any performance loss due to the use of checkpoint/restart technique. Specifically, for 20 seconds, only 505 servers (initial 504 plus one for maintenance) are needed. On the other hand, we observed that by setting the detection time at 10sec., 5sec. or 2sec. the servers are almost double from the number of the servers at

the 20 seconds. Based on the above, 20 seconds is the best choice for TCO. The trends are similar for other typical availability targets (99.9% and 99.99%).

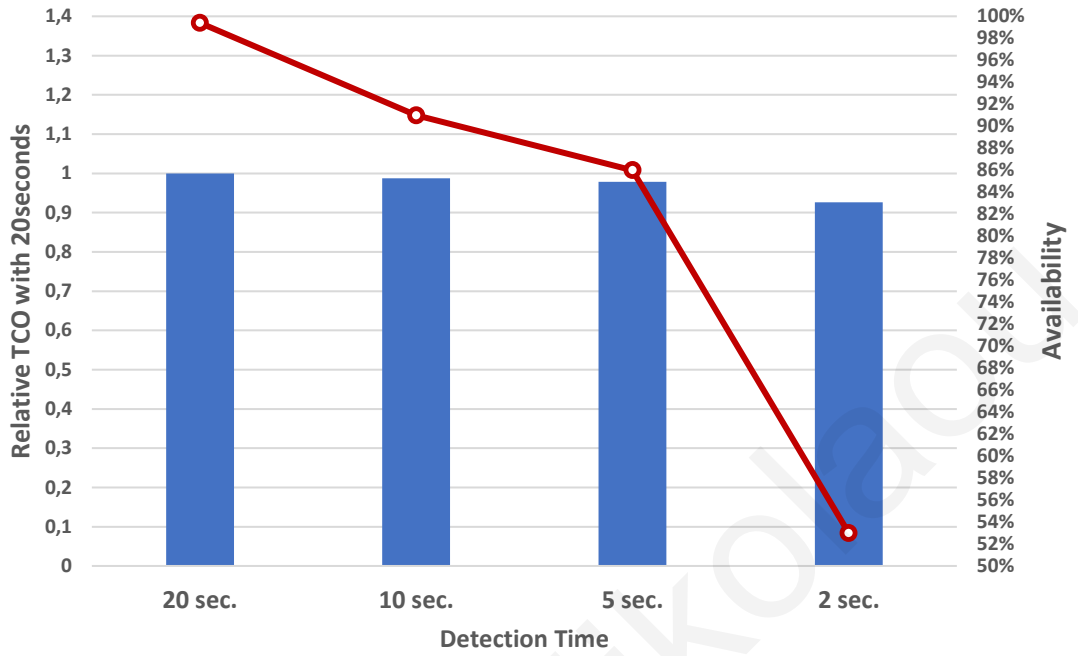

Figure 25: TCO evaluation and Availability results for different detection times

Furthermore, Figure 26 illustrates the total power consumption in watts for the whole DC. As can be seen from the figure, when the detection time is at 20 seconds, the total power is less than all the other detection times. This happens due to the fewer servers that are needed in this case.

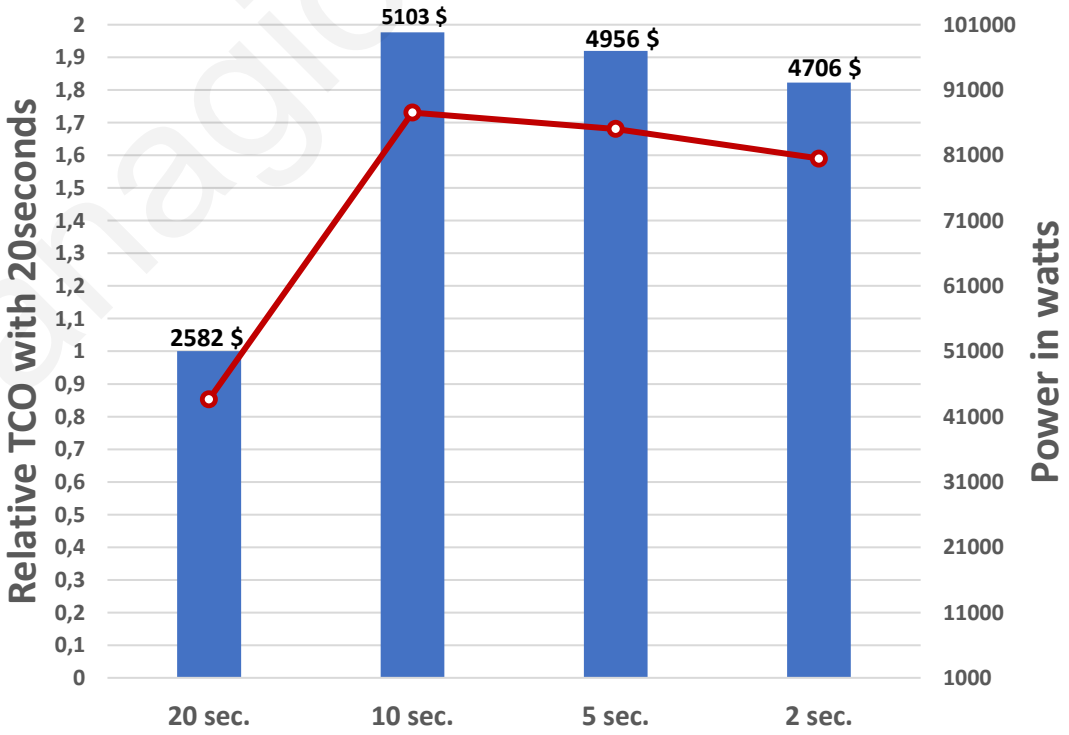

Figure 26: TCO evaluation while ensuring 99% availability

5.7 Applicability of the Key Findings

In this Section we discuss the potential uses of the findings of this work.

Our identification methodology is purely software-based. The implementation in our setup requires only 21ms to classify a workload according to the 100 different performance-counter values obtained during the first 20 seconds of the workload's execution. The clustering method has also minimal overhead (a logarithmic-function is applied to 100 values). Thus, the identification method is suitable to drive a lightweight software-based Dynamic-Voltage-Scaling (DVS) governor for under-volting a real system. The DVS governor will use our method to predict if a workload is V-low or V-high and will adjust the supply voltage accordingly. Beyond the two voltage levels we consider in this work, the DVS governor can be extended to perform CPU-Vmin predictions for additional voltage levels but this will require a more diverse and time-consuming offline CPU-Vmin characterization.

The governor can be used in systems that run various unknown batch jobs, that their execution time is not limited to few seconds. Systems that run only specific applications, such as web search and web serving, cannot benefit from the enhanced DVS governor because an exhaustive offline CPU-Vmin characterization of such applications would be sufficient to determine their CPU-Vmin and will also provide higher power savings.

We like to note that because the proposed methodology predicts a V-high for unknown workloads it can be effective in avoiding a crash from a malicious attack that accomplishes to execute a virus on a platform that causes large voltage droops such as a dI/dt virus [185]. We expect that a virus generated using a stochastic procedure, as in [185], will not match the signatures of the V-low workloads used for offline characterization. If such attacks are expected it could be wise to use the nominal supply voltage for V-high to avoid any instability or crashes due to a very powerful virus.

Chapter 6

6 Analysis of the Implications of DRAM Failures and DRAM Protection Techniques on the TCO

The pervasive economic value and societal dependence of services running in DC is made possible by their high availability and integrity. Users have come to expect these services to be virtually uninterruptible and provide correct functionality. Interestingly, the high availability and integrity perceived by users is often provided from facilities that are built using low availability commodity components [82]. This is made possible by the nature of several popular services that mainly serve numerous independent requests that are mostly read-only [4]. Furthermore, many online services require short latency and high throughput which in turn necessitates partitioning and replication of their huge working sets. Ultimately, this means that the failure of an individual DC server will typically have minimal, if any, repercussions to the service availability.

This is in direct contrast with high performance computing centers that run parallel programs with across thread data dependencies where a single server failure may result in a disruption of the entire infrastructure [83]. This problem is particularly acute for the upcoming exascale systems with a huge number of components that require extreme level of reliability from individual components to prevent frequent entire system interruptions [84].

This is not to say, however, that DC servers do not need protection from faults. For instance, an attempt to remove protection from DRAM in Google servers resulted in a subset of queries returning random documents due to a memory error that could not even be detected[4]. Consequently, DC servers employ a combination of hardware and software techniques to accomplish the desired level of availability without compromising quality of service [5].

This Chapter proposes a framework[1] , called AMPRA, for analyzing the implications of DRAM failures and DRAM protection techniques on the TCO of a DC. DRAM failures and memory protection have received a lot of attention recently with several studies showing that

[1] (Publicly Available- AMPRA: Analyzer of TCO Implications of Memory Failures and Memory Protection http://www2.cs.ucy.ac.cy/carch/xi/ampra tco.php)

DRAM is one of the main culprits for machine crashes and component replacements in today's DC and large supercomputers [42][85][86]. DRAM failure rate per server is a growing problem because despite the decrease in the failure rate per bit, one order of magnitude every 6 years, memory DIMM slots per server have been growing, to accommodate larger workings sets, leading to an overall increase in the failure rate per server [87][88][89].

To prevent systems from failing more frequently due to DRAM errors, data in memories are stored as error-correction-codes (ECC) using extra memory chips. Moreover, memory controllers are continuously upgraded to support stronger ECC capable of detecting and correcting more faulty bits. Memory controllers found in processors today [73][74] support various memory protection techniques that datacenter designers need to select from when configuring their servers.

In this work we argue that it is not straightforward to decide which DRAM protection scheme is best for a given DC setup. This challenge stems from the cost-benefit trade-off of each protection scheme with each offering a distinct combination of reliability, power, temperature, performance and server overprovisioning. Server overprovisioning is needed to: (i) ensure peak throughput in the presence of errors since some servers may need to be offline until they are repaired or replaced, and (ii) compensate for possible performance degradation due to the protection scheme used. Furthermore, the specific cost-benefits may vary depending on the service characteristics, such as memory intensity, sensitivity to collocated services and the service overall DC utilization. These and other parameters, to be identified later, are used as inputs to the framework we propose in this work to determine what is the best–in terms of TCO–memory protection scheme for a given DC.

A recent paper [30] has investigated the benefits, for a DC running web services, from the use of a hypothetical heterogeneous memory protection scheme that employs in the same server fully tested DIMMs and less tested (less costly) DIMMs. The system minimizes failures by mapping memory pages of an application to DIMMs according to their potential to cause user visible data corruption. The main focus in [30] is the data corruption analysis and not, as in our work, on a generic framework for assessing holistically the TCO implications of memory faults and protection.

6.1 Background on Memory Reliability

6.1.1 Memory Errors (Types and Metrics)

Memory errors can be categorized into transient and permanent errors, depending on their duration. Transient errors can cause reading incorrect memory values, until the faulty memory location is overwritten. Permanent errors can cause physical damage and the faulty memory location can consistently return incorrect values [90]. Therefore, to detect and correct errors, memories typically include reliability features such as ECC. Depending on the ECC strength and the type of error, an error can be correctable (CE), detectable but uncorrectable (DUE) and non-detectable (NDE) [91]. A CE error can be detected and corrected by the ECC. A DUE error can be detected by the ECC but cannot be corrected. There are many ways to recover from transient DUE errors such as to reboot the process or the whole server. Permanent errors, on the other hand, can lead to repeated DUE or CE and can be tracked by an error monitoring system that places the faulty server offline for further diagnosis and possible replacement. An NDE error is not detected by the ECC. This error may cause system crash, hang, SDC, or may be benign if the erroneous data does not affect program output [30][92]. System crashes and hangs can be detected by system software or server health monitors.

A variety of reliability metrics are used to express the occurrence of the various types of errors and capture their implications. FIT rate metric represents the number of failures in 10^9 hours. The mean time to experience an error is referred to as MTTF. Another important metric is the MTTR which can vary depending on the recovery technique. For example, rebooting a whole server usually takes more time than restarting a process. Finally, a very important metric is the availability. Availability is the probability of a system operating correctly and is usually reported in number of nines. Availability is given by the well-known equation:

$$\boldsymbol{Availability} = \frac{\boldsymbol{MTTF}}{\boldsymbol{MTTF} + \boldsymbol{MTTR}}$$

6.1.2 DRAM Error Protection

DRAM is protected from errors using extra devices per DIMM to store ECC codes. Typically, codes today use 8/16 ECC bits to protect 64/128 data bits. For example, a DDR3 memory channel is 72 bits wide with each memory channel supporting one or more DIMMs.

A single DIMM consists of multiple DRAM devices where all or a subset of them operate together to provide 72 bits. Each device can provide 4, 8 or 16 bits (referred to as x4, x8 or x16, devices respectively). For example, a 72-bit codeword can be produced using 18 x4 DRAM devices (16 devices for data and 2 for ECC) or by using 9, x8 devices (8 devices for data and 1 for ECC). To produce a 144-bit codeword requires either two bursts from the same channel or one burst across two channels. The encoding and decoding of ECC codes are usually performed in the memory controller of a processor. A processor may support various ECC options, with distinct code strength and overheads, one of which is selected at boot time. Below, we describe three commonly used memory protection ECC codes that are analyzed in this work.

Single Error Correction-Double Error Detection (SECDED) [45][46] corrects all single bit errors and detects all two-bit errors using a 72 bit codeword (64 data bits and 8 ECC bits). Many triple bit errors can be detected by SECDED code as DUE but some are miscorrected and become NDEs. Also, most of the four-bit errors are detected as DUE but some of them are NDEs. SECDED can be used for both x4 and x8 devices but the power consumed by x8 devices is lower because they can provide the same capacity with fewer devices [73][93].

Single-Chip error correction and Double-Chip error detection or Chipkill [47], is commonly used for DRAM protection in high availability servers and large-scale systems because it has the ability to correct all the errors that appear in a DRAM device and to detect errors in two DRAM devices. Chipkill relies on symbol-based coding to perform error detection and correction. In a symbol-based code, each codeword is composed of multiple symbols, with each representing a group of bits. There are various flavors of Chipkill with different strengths and overheads. Modern processors usually support Chipkill with 16 ECC bits for 128 data bits that are interleaved across two DIMMs placed in two channels [72] [73][74]. This Chipkill implementation corrects all the errors in a single device and detects all the errors in two devices [72]. It uses standard DDR3 with burst length of 8, to read two 64B blocks per access one of which is wasted for systems with 64B cache block size. Consequently, Chipkill can waste bandwidth, hurt performance and increase energy consumption [75][76][77]. To limit reading one 64B block per memory access, burst-chop is used to reduce the burst length from the usual eight down to four [76][94]. Although burst chop can be used to save the energy of four bursts, the access time cannot be reduced and the bandwidth is still wasted [75][76]. We refer to this dual channel Chipkill technique as ChipkillDC.

Another Chipkill implementation, similar to ChipkillDC, uses only a single channel to provide 16 ECC bits for 128 data bits [74]. This Chipkill implementation encodes and

decodes a codeword every two bursts and it is able to correct all the errors in a single device and detect 99.99999963% of the errors in two devices [72]. We refer to this single channel Chipkill technique as ChipkillSC. The ChipkillDC per burst access is wider than ChipkillSC, since it accesses twice the number of devices per burst (e.g. 36 vs 18 with x4 devices). This enables slightly better reliability than ChipkillSC but consumes more power and wastes bandwidth which can hurt performance. On the other hand, ChipkillSC can also degrade performance because it needs to wait for two bursts to form a codeword. Both Chipkills provide superior reliability as compared to SECDED but a x8 SECDED protected DRAM can be more power efficient than a x4 Chipkill implementation. Evidently, each protection scheme has its own pros and cons, motivating the development of a framework to analyze and decide which DRAM protection scheme to use for a given setup.

6.1.3 Datacenter's Reliability and Availability

The reliability and availability target of a service running on a DC is accomplished through a combination of hardware and software mechanisms and policies. This also aims to ensure satisfying the quality of service (QoS) requirements even in the presence of errors and downed servers [4][5]. These mechanisms typically rely on hardware and software-based detection and software-based error management.

Errors can be handled at the application and software level [95][96]. OS can deal with all the uncorrectable memory errors detected by ECC or diagnosed after observing system anomalies, such as machine crashes and hangs. One response to a diagnosed permanent memory error is to replace the faulty DRAM component or even the whole server module. Unfortunately, this is costly but, in some cases, inevitable. Another less costly recovery action is to use page retirement. Page retirement removes a physical memory page that experiences repeatedly errors [97], but may not be effective for coarse errors affecting many pages. Process/server reboot, another recovery option, works well in the case of transient uncorrectable errors.

6.1.4 Online, Offline Services and Co-location

Online services are interactive services that perform significant processing over big datasets and are driven from a huge number of user requests. Because of their interactive nature, these services require responsiveness in the order of hundreds of milliseconds and have high QoS requirements [98]. A concrete example of an Online service is the Web Search [99].

Web Search runs on thousands of servers to provide high throughput. Typically, the service working set is replicated and partitioned across many servers to ensure low latency but also to achieve high availability. For example, when a server fails, during its repair time the job that runs on it can be restarted on another server with minimal repercussions on the QoS and availability of the overall service. Web Search must provide high levels of availability, such as four nines (0.9999), and is overprovisioned with extra servers to deal with various hardware failures.

Offline services are non-interactive and do not have strict QoS constraints, e.g. response latency. Examples of such services are: Data Analytics, file backup, image processing, video compression, optimization search, and simulation cycles. Online and Offline services can be run together (collocated) to improve server and DC utilization [64][65]. Specifically, when an Online service cannot use all the available cores in a server concurrently, due to QoS constraints, some or all of the remaining cores can run some Offline services. This must be done without affecting the QoS of the interactive service. In this work we show that the characteristics of collocated services can influence the choice of memory protection that is best for TCO. In fact, the experimental data show that selecting the memory protection based only on what is best for the Online service can be suboptimal for the overall DC TCO.

6.1.5 Total Cost of Ownership (TCO)

Several models have been proposed for guiding DC design TCO such as [80][100][101][102][103][104]. These models either do not account for the effects of failures or only treat them in a cursory manner.

A recent very relevant work [30], performs software fault injection campaigns in DRAM to characterize the SDC rates of web services. While the SDC analysis in [30] is seminal, and can be useful for one of the inputs of our framework, its TCO analysis, as well as that of previous TCO efforts, lack the following capabilities provided in our work: explore advance ECC schemes, consider the implications of multi-bit errors, account for the performance, power and temperature implications, consider the ramifications of collocated services, measure DC TCO operational and capital expenses (not only server cost) and account for failing module replacements and maintenance policy. We investigate the significance of analyzing these parameters later in this work. Additionally, this work provides a detailed description of the framework to facilitate its adaptation and use, and it is capable to reveal TCO optimization opportunities even for a DC in use today.

6.2 AMPRA Framework

A framework for assessing the implications of DRAM errors and protection techniques on the TCO of a DC has been developed. The proposed framework input parameters, processing components and the flow of information are shown in Figure 27. As far as we know, this is the first framework that attempts to combine all these variables together and eventually produce the TCO of a DC. The framework consists of eight different models: Energy, DIMM Cost, DRAM FIT, Availability/MTTF, SDC Derating, Performance, Thermal and TCO model. The inputs of each model are shown inside dashed boxes, while the outputs are presented with grey color. The final outputs of the framework, TCO and System Reliability (MTTF SDC), are presented with dark red color. Next, we elaborate in more detail the inputs, the flow through the framework and its intermediate and final outputs.

6.2.1 Performance Model

The performance model takes as inputs the server configuration, DRAM ECC technique and the number of threads per server running Online and Offline services. The goal of this model is to determine the performance of the Online and Offline services when running alone and collocated. The performance model also facilitates the comparison between two DCs by taking as input the server performance of a reference DC and producing the Performance Degradation (PD) of another DC relative to the reference DC. PD is used subsequently, by the TCO model, to determine the extra overhead, e.g. additional/fewer servers, needed by a DC to match the performance of the reference DC. The difference in the DCs server performance can be due to the choice of ECC technique or the services they run in this work. The PD is used by the TCO model to estimate the extra servers are needed (called hot spares) to compensate the performance degradation. For instance, if the expected performance is at 90% of the maximum and the workload requirements are 10000x throughput (e.g. 10000 cores running separate threads), then we will need (10000/0.9 - 10000) 1111 extra cores to meet application's requirements, which translates to extra server costs for acquisition, maintenance, power consumption and space. PD takes values from 0 to 1, with 0 meaning no degradation at all and 1 means no operation. The performance is thus given by $1 - PD$.
It is critical to quantify the performance of different ECC techniques since ChipkillDC, the technique with the strongest code, can incur up to 38% performance overhead compared to SECDED for memory intensive workloads [105] due to its wasteful use of bandwidth. Similar observations have been made in several other studies [76][77][105][106].

Performance for a given ECC can be determined using simulations or real hardware. Simulations are useful in the absence of hardware that supports the required ECC modes and for exploring new ECC techniques. In this work we determine the ECC performance implications with real hardware where services are deployed. Besides the protection technique, performance can also be sensitive to the type of address interleaving used. State of the art server processors, support various options of interleaving: Full interleaving, Channel interleaving and No interleaving [78]. ChipkillDC always uses full interleaving, because it splits a cache line across two DIMMs in different memory channels [107]. Although, the address interleaving option may matter for ChipkillSC and SECDED performance, we only consider full interleaving in this work and leave the exploration of the interleaving impact for future work.

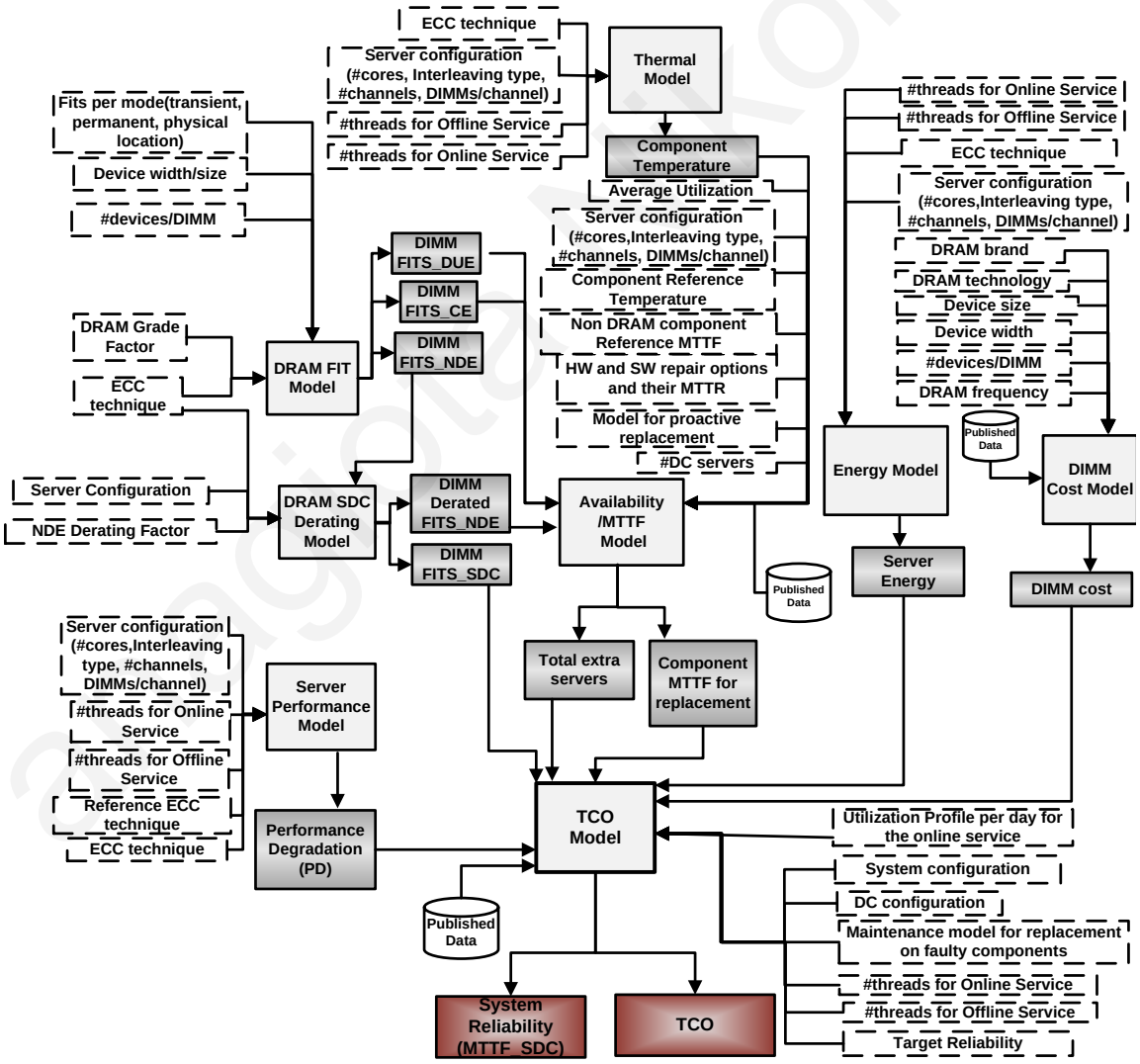

Figure 27: AMPRA Framework Parameters, Components and Information Flow

Apart from the performance differences caused by the choice of the ECC technique, performance of an Online service can be affected by the Offline service that it is collocated with. In particular, if due to QoS constraints not all the cores of a server run Online services, then Offline services can be run concurrently in the unused cores. Co-location can improve a machine's utilization, but this should come with a minimal impact on the QoS of the Online services due to interference from shared resources [64]. Performance Degradation (PD) is defined as follows:

$$\boldsymbol{PD = 1 - \frac{Time_{OnlineA}}{Time_{(Online+Offline)B}}}$$

where $Time_{Online\ A}$ is the time to run the Online service on a server using ECC protection technique A, and $Time_{(Online+Offline)B}$ is the time to run the Online service when collocated with an Offline service on a machine with ECC protection technique B. The PD is used in this work to evaluate the following three scenarios: (i) compare DCs running only the same Online service (no co-location) but use different ECC techniques, (ii) compare DCs using the same ECC technique (A=B) and running the same online service but with the one DC running collocated jobs and the other not running them, and (iii) same as scenario (ii) but with each DC using different ECC technique (A≠B).

The PD can be augmented to account also, for other performance metrics, such as the tail response latencies (90th, 99th) for the Online service. More specifically, the framework can be augmented to assess the profit depending on the response latency [108].

6.2.2 Energy Model

Besides their performance implications, ECC techniques and co-location can also cause an increase in the energy consumption of a server. The Energy model is used to determine the energy of a given server for two use cases: a high utilization scenario when the server is running Online and Offline services together and a low utilization where the server is running only the Offline service. For DCs with no collocated jobs (no offline jobs) the low utilization case corresponds to the idle energy. The Energy of a server can be determined analytically, using simulation or by using either hardware or software energy monitoring tools [27].

The per server energy is used by the TCO model to estimate the overall energy of a DC based on a server utilization profile of a DC.

6.2.3 Thermal Model

Component lifetime reliability has a strong temperature correlation. Moreover, temperature in processors and DRAM depends on power density which itself depends on processor performance and memory utilization. Therefore, we have developed a thermal model to determine the temperature of different server components, such as CPU and DRAM, to capture the interaction between lifetime reliability with the memory protection used by a server and the mix of services it is running.

The temperature of a server component, can be determined by using analytical model measurements, such as 3D-ICE [109], or real hardware measurements using thermal sensors and hardware-specific software such as lmsensors [79]. We measure the temperature per server component, for each specific ECC technique and use case. The component temperature is then used by the Availability/MTTF model to evaluate the effects of temperature on MTTF, component replacements and server overprovisioning.

6.2.4 DRAM FIT Model and Modeling DRAM Grades

The DRAM FIT model is used to produce the raw FIT rates for CE, DUE and NDE errors per DIMM. The model inputs are: a specific ECC protection technique for a given DIMM configuration (number of devices, their size and width), the fault rates per device for various failure modes (row, column, bank, etc.), failure types (transient or permanent) and number of faults (1, 2 etc.). The failure rates can be produced analytically using either projected rates and failure distributions or rely on failure rates obtained in field studies of DRAM errors.

Analytical failure models have been developed, based on probabilities for spatial errors, and failure rates from a large scale field study of DDR3 memory [41]. For SECDED the probabilities are obtained for a given number of faulty bits whereas for both Chipkills they are obtained for a given number of faulty symbols.

The developed analytical equations to calculate the different FIT rates for ChipkillDC and ChipkilSC are shown below:

ChipkillDC equations

$$CE = P_{fail1dev} \sum_{1}^{x} P_x{}^1 (1 - P_x)^{n-1}$$

$$DUE = P_{fail2dev} \sum_{1}^{y} \sum_{1}^{x} (DF_x \binom{n}{1} P_x{}^1 (1 - P_x)^{n-1} * \binom{n-1}{1} P_y{}^1 (1 - P_y)^{n-2})$$

$$NDE = P_{fail3dev} \sum_1^z \sum_1^y \sum_1^x (DF_x \binom{n}{1} P_x{}^1 (1-P_x)^{n-1} * DF_y \binom{n-1}{1} P_y{}^1 (1-P_y)^{n-2} * \binom{n-2}{1} P_z{}^1 (1-P_z)^{n-3}) + P_{fail4dev} \sum_1^j \sum_1^z \sum_1^y \sum_1^x (DF_x \binom{n}{1} P_x{}^1 (1-P_x)^{n-1} * DF_y \binom{n-1}{1} P_y{}^1 (1-P_y)^{n-2} * DF_z \binom{n-2}{1} P_z{}^1 (1-P_z)^{n-3} * \binom{n-3}{1} P_j{}^1 (1-P_j)^{n-4})$$

ChipkillSC equations

$$CE = P_{fail1dev} \sum_1^x P_x{}^1 (1-P_x)^{n-1}$$

$$DUE = (P_{fail2dev} \sum_1^y \sum_1^x (DF_x \binom{n}{1} P_x{}^1 (1-P_x)^{n-1} * \binom{n-1}{1} P_y{}^1 (1-P_y)^{n-2})) * \text{ProbFail}$$

$$NDE = (P_{fail3dev} \sum_1^z \sum_1^y \sum_1^x (DF_x \binom{n}{1} P_x{}^1 (1-P_x)^{n-1} * DF_y \binom{n-1}{1} P_y{}^1 (1-P_y)^{n-2} * \binom{n-2}{1} P_z{}^1 (1-P_z)^{n-3}) + P_{fail4dev} \sum_1^j \sum_1^z \sum_1^y \sum_1^x (DF_x \binom{n}{1} P_x{}^1 (1-P_x)^{n-1} * DF_y \binom{n-1}{1} P_y{}^1 (1-P_y)^{n-2} * DF_z \binom{n-2}{1} P_z{}^1 (1-P_z)^{n-3} * \binom{n-3}{1} P_j{}^1 (1-P_j)^{n-4})) + (DUE * (1 - ProbFail))$$

, where **n** is the number of devices, **x,y,z,j,** are the type of errors (single word, single bit, single column etc.) where each of them can take all the 14 values from [41], **P** is the probability of a device failure due to one of the 14 different types of errors, $\mathbf{P_{failxdev}}$ is the probability of an n-device DIMM to experience x device errors (eg. $\mathbf{P_{fail2dev}}$ =Probability of a DIMM to experience 2 errors), **DF** is the derating factor and **ProbFail** is the probability of the ChipkillSC to experience a failure. ChipkillSC implementation encodes and decodes a codeword every two bursts and it is able to correct all the errors in a single device and detect 99.99999963% of the errors in two devices. Thus, the **ProbFail** is equal to 0.9999999963.

CE for both Chipkills can be estimated by summarizing all the combinations of the probabilities for different fault types (such as single word, single bit etc.) and multiplying these probabilities with the probability to have a single fault in one device. DUEs for ChipkillDC can be estimated by summarizing all the combinations of the probabilities of different fault types that happen in two devices in a single codeword. On the other hand, for ChipkillSC DUEs are then multiplied with the ProbFail to not account for the faults that cannot be detected in a single codeword.

To determine the probability of two faults occurring in the same codeword in both Chipkills, we use a derating factor (DF) to derate each probability for each unique fault type combination on two devices. For example, the DF for single column faults in a device is $\frac{1}{number\ of\ columns}$. Finally, NDEs for ChipkillDC can be estimated by summarizing all the combinations of probabilities for different fault types that happened in three and four devices in a single codeword. For ChipkillSC, also, the additional number of DUEs that is not included in the DUE equation is added to the estimation of NDEs ($DUE * (1 - ProbFail)$). The probability of three or four faults happening in the same codeword is determined by derating appropriately each unique fault type combination on three and four devices. We also, derate each combination with an appropriate factor (DF) to account for the likelihood of a fault combination happening in the same codeword. Each combination contributes to a different repair action depending on whether it includes only transient errors or it has at least one permanent error.

SECDED equations are based on an iterative process that goes through all the bits and estimates the probability of having single bit, double bit, triple bit and quadruple bit errors.

Each equation computes the probability for all device combinations that can produce a given number of faults.

The various FIT rates, derived from the above analytical equations, for all the ECC schemes considered in this work are shown in Table 11.

Table 11: FIT rates of transient (Tr.) and permanent (Pr.), CE, DUE and NDE errors for each protection technique FIT/device

	Correctable (FITS_CE)		Uncorrectable (FITS_DUE)		NDE (FITS_NDE)			
	Tr.	Pr.	Tr.	Pr.	Tr.		Pr.	
ChipkillDC	19.925	20.405	$1.61*10^{-4}$	$5.53*10^{-4}$	$1.52*10^{-16}$		$1.81*10^{-15}$	
ChipkillSC	19.924	20.404	$1.66*10^{-4}$	$5.65*10^{-4}$	$6.13*10^{-13}$		$2.09*10^{-12}$	
					(FITS_MCE)	(FITS_UDE)	(FITS_MCE)	(FITS_UDE)
x4SECDED	17.13	16.99	2.72	3.32	0.0639	0.00537	0.0841	0.0074
x8SECDED	34.26	33.98	5.44	6.65	0.1279	0.0107	0.168	0.0148

Each row of the table depicts the FIT rates for different protection schemes for Correctable transient (FITS_CE-Tr.), Correctable permanent (FITS_CE-Pr.), Uncorrectable transient (FITS_DUE-Tr.), Uncorrectable permanent (FITS_DUE-Pr.), Non-detectable transient (FITS_NDE-Tr.) and Non-detectable permanent (FITS_NDE-Pr.) errors.

SECDED NDEs are divided further into two categories: Miscorrectable errors (FIT_MCE) and Undetectable errors (FIT_UDE). For SECDED, the analytical equations consider up to four faulty bits per codeword (since probabilities for more faulty bits are typically extremely low). Therefore, miss-corrections, FIT_MCE, occur when three bits are flipped and the syndrome value is valid. Assuming the SECDED code in [46], the probability to have miss-correction due to triple bit errors is 56.28%. Undetected errors, FIT UDE, occur when four bits are faulty and the syndrome value is zero. This probability equals to 0.8%. We consider these two extra categories only for SECDED because for both Chipkill schemes the NDE FIT rates are extremely small. Table 11 also, presents fault rates for SECDED with x8 devices assuming that they are two times bigger than x4 devices. Since [41] does not provide raw fault rates for x8 devices we double the FIT rates of x4 devices. This effectively assumes no fault overlapping (two different faults happening on the same bit(s)). This is a reasonable assumption for the fault rates in this work since the probability of overlapped faults is extremely low, 10^{-15}. We optimistically assume that there are no multibit errors with more than four bits with x8 devices.

DRAM Grades: One other FIT model parameter is the DRAM grade. The DRAM grade attempts to capture the variation in DRAM quality [110] with lower grade DRAM experiencing less failures. It is expected that lower grade DRAMs cost less[110][111][112], thus presenting an opportunity for trading-off reliability for TCO reduction. The DRAM grade is expressed as a numeric factor that is used to multiply the raw fault rates in [41] (i.e.

the larger the DRAM grade factor the higher the failure rates). The choice of range of factors considered is hypothetical and aims to explore how big of an opportunity DRAM grades present for TCO optimization.

6.2.5 DRAM SDC Derating Model

DRAM SDC Derating Model is used to estimate, for each ECC technique, the fraction of NDE FITs per server that lead to SDC errors, i.e. affect the service output without been detected. We refer to this fraction as FITS_SDC. This can be determined by either characterizing a service using fault injections [30] or with analytical models that consider the dynamic access patterns to different entries of various data structures of a service (hash tables, lists etc.). In this work we simply examine the FITS_SDC trends for a range of derating factor values to help determine the robustness against SDCs of the various ECC techniques. The model accuracy can be improved by performing fault injection or analytical modeling of specific services.

The FITS_SDC per server is used by the TCO model to estimate the MTTF_SDC for the whole system (for all the servers), a metric of the service reliability. The NDE FITs that do not cause SDCs (Derated FITS_NDE =FITS_NDE – FITS_SDC) are pessimistically assumed to cause some visible user/system failure (i.e. none is benign) and are provided to the Availability/ MTTF Model that estimates the total number of extra servers needed to ensure peak throughput in the presence of these failures.

6.2.6 Availability/MTTF Model

The Availability model takes as inputs many parameters: the DRAM FIT rates for CE, DUE and derated NDE for a given ECC scheme (provided by the FIT and SDC models), the number of DIMMs/server, the initial number of DC servers, the actual temperature of each component (provided by the Thermal model), the reference temperature and MTTF of each component, the DC utilization, the policy for proactive replacement and the different hardware and software repair techniques with their MTTR values (Table 12).

Table 12: MTTR for various repair actions due to different types of failures

	Details	Time
MTTRDIMM_rpl	Replace DIMM	1440 min
MTTRpg_r	Page retirement	100 min
MTTRrbt	Server reboot	100 min
MTTRecc	ECC	0 min

We consider various repair techniques: ECC protection, page retirement, server reboot and DIMM replacement. ECC can be used to repair transient correctable errors. Page retirement can be used to repair permanent correctable errors. Server reboot can be used to repair transient uncorrectable errors. Finally, DIMM replacement can be used to repair uncorrectable permanent errors.

The Availability/MTTF model estimates the MTTF for a permanent failure for each server component (CPUs, DRAM DIMMs, disks) using the Arrhenius equation [113] to compute the MTTF acceleration factor (K) depending on the component actual temperature, provided by the thermal model, and the component reference temperature and MTTF, obtained from published data sheets. Note that the DRAM MTTF acceleration due to temperature is performed only for permanent correctable and uncorrectable faults.

The final MTTF of each component is calculated based on the DC average utilization. For example, for 20% average utilization the final MTTF is the weighted harmonic mean of the MTTF with peak activity (Online and Offline co-running) for 20% of the time and the MTTF with low activity (running only the Offline service) for the rest 80% of the time. The final MTTF of each component is then used by the TCO model to determine the number and cost of the extra DIMMs (N_{DIMM_rpl}), CPUs (N_{cpu_rpl}), and Disks (N_{disk_rpl}) needed for replacement. The MTTFs are also used, together with the MTTRs, to determine the total number of extra servers that are needed to ensure peak throughput in the presence of failures and some server unavailability. Table 12 lists the duration for each repair technique (MTTR). We assume that ECC correction has a negligible MTTR. We have checked a range of values for reboot and page retirement and do not observe much sensitivity due to the rarity of these events. It should be noted that the model is not specific to the techniques and the repair times that are shown in Table 12, other repair schemes and MTTRs can be added to the model.

To compensate the performance loss due to the time required to repair faulty DIMMs, we need to overprovision the DC with extra servers. The following equations are used to

calculate the extra servers to cover the performance loss due to: page retirement repairs (N_{pg_r}) and server reboot repairs (N_{rbt}). The N_{ecc} is zero because $MTTR_{ecc}$ is negligible.

$$N_{pg\_r} = \left(1 - \frac{\frac{MTTF_{pr\_CE}}{\#DIMMS}}{\frac{MTTF_{pr\_CE}}{\#DIMMS} + MTTR_{pg\_r}}\right) * (N_{srvmodulesreq})$$

$$N_{rbt} = \left(1 - \frac{\frac{MTTF_{tr\_DUE}}{\#DIMMS}}{\frac{MTTF_{tr\_DUE}}{\#DIMMS} + MTTR_{rbt}}\right) * (N_{srvmodulesreq})$$

where the $N_{srvmodulesreq}$ is the initial number of server modules required for the peak workload assuming no failures, and #DIMMS is the number of DIMM slots per server module. The $MTTF_{pr\_CE}$ for page retirement is given by $\frac{10^9}{FITS_{pr\_CE} * \#devicesPerDIMM}$, where #devicesperDIMM is the number of devices per DIMM and the $MTTR_{tr\_DUE}$ for server reboot is obtained using $\frac{10^9}{FITS_{tr\_DUE} * \#devicesPerDIMM}$.

We assume that page retirement and reboot events never happen together on the same node because this occurs with very low probability for the rate of failures we considered. Finally, the total number of extra servers $Total_{extra\_servers}$ needed to make up the performance loss due to memory module repairs, is determined by:

$$Total_{extra\_servers} = \lceil N_{pg\_r} + N_{rbt} \rceil$$

The MTTF of a $DIMM_{replacement}$ due to uncorrectable permanent errors, $MTTF_{pr\_DUE}$, is calculated using the following equation:

$$MTTF_{pr\_DUE} = \frac{10^9}{FITS_{pr\_DUE} * \#devicesPerDIMM}$$

To limit the effects of uncorrectable errors we explore the benefit of employing proactively replacement. In particular, when "proactive replacement on correctable errors" is used, the DIMM replacement is applied on every permanent correctable error. As reported before, it is difficult to predict from correctable errors the uncorrrectable error rate, but the proactive replacement on every correctable error can help minimize the uncorrectable error rate by

70% [42][86]. The $MTTF_{pr\_DUE+pr\_CE}$ for a DIMM replacement with proactive replacement is given by:

$$MTTF_{pr_{DUE}+pr_{CE}} = \frac{10^9}{(FITS_{pr_{DUE}} + FITS_{pr_{CE}} * \#devicesPerDIMM}$$

6.2.7 DIMM Cost Model

The DIMM cost is determined by six parameters: DRAM brand, DRAM technology, device width (x4 or x8 devices), device size, the number of devices per DIMM and DRAM frequency. The DIMM cost is estimated with the help of publicly available data and price listings.

6.2.8 TCO Model

The last component of the framework is the TCO Model that is used to estimate the cost of a DC. The model is based on the TCO tool by [80] extended to take as inputs the outputs produced by the various models in our framework plus some other parameters, such as the utilization profile per day for the online service, DIMM FITS SDC and the target reliability. The performance degradation (PD) determines the number of extra spares to compensate the performance loss due to ECC overheads or service co-location:

$$\boldsymbol{N_{hotspares} = \frac{N_{srvmodulesreq} + Total_{extra\_servers}}{1-PD} - (N_{srvmodulesreq} + Total_{extra\_servers})}$$

where $N_{srvmodulesreq}$ is the number of server modules required for the peak workload without failures, and $Total_{extra\_servers}$ is estimated by the Availability/MTTF model and accounts for the number of extra servers needed to make up the performance loss due to the repair techniques. In the case that PD is negative then $N_{hotspares}$ returns a negative number. This means performance improves relative to reference DC and fewer servers are required. Consequently, the total number of servers in a DC becomes:

$$\boldsymbol{N_{hotspares} = N_{srvmodulesreq} + Total_{extra\_servers} + N_{hotspares}}$$

The above is used as an input to the TCO model that repeats the TCO estimation until the $N_{svrmodules}$ converges (empirically, at most three iterations are needed.). Replacement on an

error can be performed at different granularities, e.g. faulty component or the whole server. In our model, we apply component replacement. The number of spares needed for replacement of faulty components are estimated for each component type (DIMM, disk and CPU) as shown below:

$$N_{DIMM\_rpl} = \frac{N_{svrmodules} * \#DIMMS}{MTTF_{dimm} + MTTR_{DIMM\_rpl}}$$

$$N_{disk\_rpl} = \frac{N_{svrmodules} * \#DISKS}{MTTF_{disk} + MTTR_{disk\_rpl}}$$

$$N_{cpu\_rpl} = \frac{N_{svrmodules} * \#CPUS}{MTTF_{cpu} + MTTR_{cpu\_rpl}}$$

For this study the MTTF and MTTR values for the CPU and disk components are derived from public data [115]. The$MTTF_{dimm}$ is derived from the analysis provided in this work and is equal to $MTTF_{pr\_DUE}$ unless proactive replacement is employed in which case is given by $MTTF_{pr\_DUE+pr\_CE}$. One other input to the TCO model is the utilization profile per day for the Online service. This parameter captures the dynamic load behavior of a DC running a specific Online service. Different utilization profiling graphs are available from various studies [4][121]. In our analysis, we use the utilization graph from [4]. Utilization affects the DC energy and replacements and consequently the total TCO. Finally, the TCO model checks whether the System Reliability (MTTF_SDC) satisfies a given target reliability. The system MTTF_SDC, is given by:

$$MTTF_{SDC} = \frac{MTTF_{SDC\_server}}{N_{srvmodules}}$$

where $MTTF_{SDC\_server}$ is the total MTTF for SDC errors for a server component and it is calculated based on the DIMM FITS_SDC and #DIMMS.

6.3 Experimental Methodology and Models Assumptions

Here, we describe some implementation details for each framework model as used in our experiments.

FIT and Availability model: The input raw fault rates of the FIT model are based on the normalized data reported in [41] converted to absolute rates, for each protection technique,

using the equations in Section 6.2.4 and the single-bit probability projected in ITRS [122]. In our analysis, we considered 4GB rank with x4 DRAM devices and 2Gb per device, and also x8 DRAM devices with 4Gb per device. To provide the FIT rates for larger x8 devices we double the FIT rates of Table 11.

Performance model: To evaluate the performance overhead of ChipkillDC relative to ChipkillSC, we use a dual socket Intel Xeon E5620 system with the configuration shown in Table 13.

Table 13: Server and main memory configuration

Number of CPUs	2
CPU	Intel Xeon E5620 2.4GHz
Number of cores per CPU	4
DRAM technology	DDR3
Channels per CPU	2
DIMMS/channel	1
Ranks/DIMM	2
DRAM device	x4
DIMM capacity	8GB
Turbo mode	Disabled
Level of Interleaving	Full interleaving

We experimented using a 2.4GHz processor frequency with disabled turbo mode. To measure the performance degradation of ChipkillDC, the server memory system is set in "Lockstep Mode". This mode combines together two DIMMs from different channels to form a 144 bit codeword [78][107][123][124]. To measure the performance degradation of ChipkillSC, the server memory system is set in "Advance ECC Mode". This mode uses a single channel and combines two bursts to form 144-bit codeword. Both modes are set by accessing the BIOS through a BIOS Serial Command Console interface (CLI) [78].

Our evaluation used Web Search, an Online benchmark from CloudSuite [50][51] and different Offline benchmarks, a Data Analytics also from CloudSuite [50][51] and few others from SPEC CPU2006 [52]. We considered four different types of Offline benchmarks according to their memory behavior: Memory Intensive (MI), Mcf, Compute Intensive (CI), Gcc, Streaming (STREAM), H264ref, from SPEC CPU 2006 and Data Analytics-Map Reduce (MR), from CloudSuite, with two different model sizes (obtained from training with 500MB and 49000MB data).

To evaluate the performance overhead of Web Search benchmark, four blade servers are used: one client server with multithreaded client process where each thread run on a single core, one frontend server, one index server and one document server with a traffic of 100K queries and a dataset size of 6GB. To increase the number of concurrent requests, we

increased the number of clients. One or two instances of each Offline service run concurrently in the server that performs the index search. The performance degradation of the Online service is measured by running it first in isolation and then co-running it in combination with one or two instances of the same Offline service, for different ECC schemes and number of threads.

We run each experiment 5 times and each time we collected the average search time and the 99th tail response latency of the Online service. The results presented are calculated by removing the minimum and maximum values.

Energy and Thermal model: The energy numbers are collected using the HPiLO3 [27] which allows to monitor the power consumption of a server at a given time. The results are used to calculate the average power numbers used in the TCO tool. The HPiLO is also used to measure the temperature of DRAM and Disk components. To track the CPU temperature, we use lm-sensors [79]. We have validated the power and temperature measurements using DRAM and CPU stress test programs [125]. The reference MTTF temperature numbers for CPU and disk are 45 ◦ C and 35 ◦ C, respectively according to [126][127].

DIMM cost model: The parameters of the DRAM cost model for 8GB DIMMs are shown in Table 14 and are obtained from public data [119][120].

Table 14: Server configuration and parameters

Components	Cost ($)
2 Processors 2 Disks Other (Case, power supply & motherboard)	130 each [114] 60 each [115] 308 [116][117][118]
DRAM x4 DRAM x8	99.66 [119] 64.74 [120]
#active cores	2
Proactive replacement	disabled

TCO model: To estimate the Total Cost of Ownership (TCO) we extended the COST-ET tool proposed in [80]. Our framework is implemented as a wrapper around this earlier tool. For each experiment an initial population of 50000 server modules is used assuming an average utilization of 31% [4]. TCO results are presented assuming servers running Web Search only on two cores to meet a QoS constraint. Analysis for collocated services assumes that the default configuration runs Online services on two cores and Offline services on the remaining one or two cores (depending on the DC configuration). Our analysis evaluates

servers that have the same DRAM configuration for x4 and x8 DIMMs in terms of the number of channels, DIMMs per channel, DIMM capacity (8GB) and ranks per DIMM, to provide the same bandwidth for both configurations.

The costs, shown in Table 14, are derived from publicly available data [115][116][117][118][128]. Finally, the various parameters for the datacenter configuration shown in Table 15 are derived from [4][80][129].

Table 15: Datacenter Configuration

Cost of building	3000\$/m^2
Cost of cooling	12.5\$/W
Cost of electricity	0.07\$/W
Cooling area overhead	1.2
Network per rack	10K\$-360W [117]
Maintenance salary per rack	200\$ (monthly)
Datacenter depreciation	15 years
PUE	1.2
Server modules in Datacenter	50,000

For the baseline configuration, we select ChipkillSC memory protection technique, with index servers running Web Search in two cores without co-location. The proactive replacement on correctable errors is not employed unless specified otherwise.

6.4 Results

We investigate how the various proposed framework parameters affect the DC TCO and the choice of the DRAM protection technique. We present different case studies to assess the impact of (a) the number of DIMM slots, (b) DRAM grades (increasing fault rates), (c) ChipkillDC and ChipkillSC Performance, Energy and TCO for an Online service running alone and collocated with Offline services, (d) NDE errors on the System Correctness, and (e) sensitivity of TCO to various parameters considered in this work.

6.4.1 Implications of DRAM Capacity on TCO

This case study investigates how the TCO of each protection technique is affected by the number of DIMM slots (8GB per DIMM) per server. The results are shown in Figure 28.

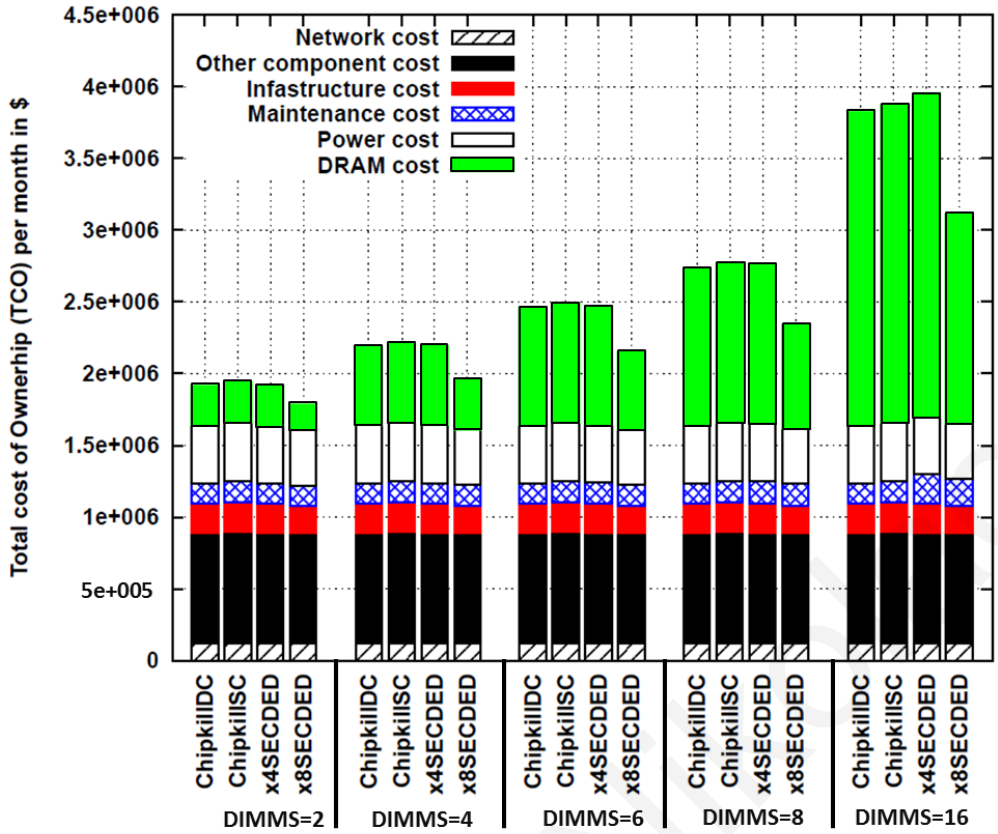

Figure 28: TCO results for different DIMMs slots

The x-axis presents different protection techniques according to the number of DIMMs per server and the y-axis shows the TCO breakdown (infrastructure, network, maintenance, power, DRAM and other server components cost such as disk, cpu, board etc.) per month in dollars. As seen in Figure 28, with increasing number of DIMM slots per server, the DC cost for all the protection techniques also increases. Also, it is observed that x8 SECDED offers better TCO as compared to the other protection techniques. The x8 SECDED benefits grow with increasing number of DIMM slots because of the lower cost and power of the x8 devices. On the other hand, as the number of DIMMs per server increases, x4 SECDED becomes the technique with the highest cost due to its lower reliability. The results in Figure 28 clearly suggest that the most cost-effective protection technique, from the four investigated in this case study, is x8 SECDED.

6.4.2 DRAM Grades and TCO

The server configuration selected for the following case studies has four DIMMs, one in each channel, with a total of 32GB DRAM per server node. As mentioned earlier, it is interesting to investigate the trade-offs of several grades of DRAM quality where better quality has higher cost and more reliable parts. Our analysis assumes a large range of DRAM grades for

better exploration of the opportunities from having DRAM products with varying cost-quality. Figure 29 shows the normalized TCO results for 20 different grades (baseline DRAM is denoted by GradeA).

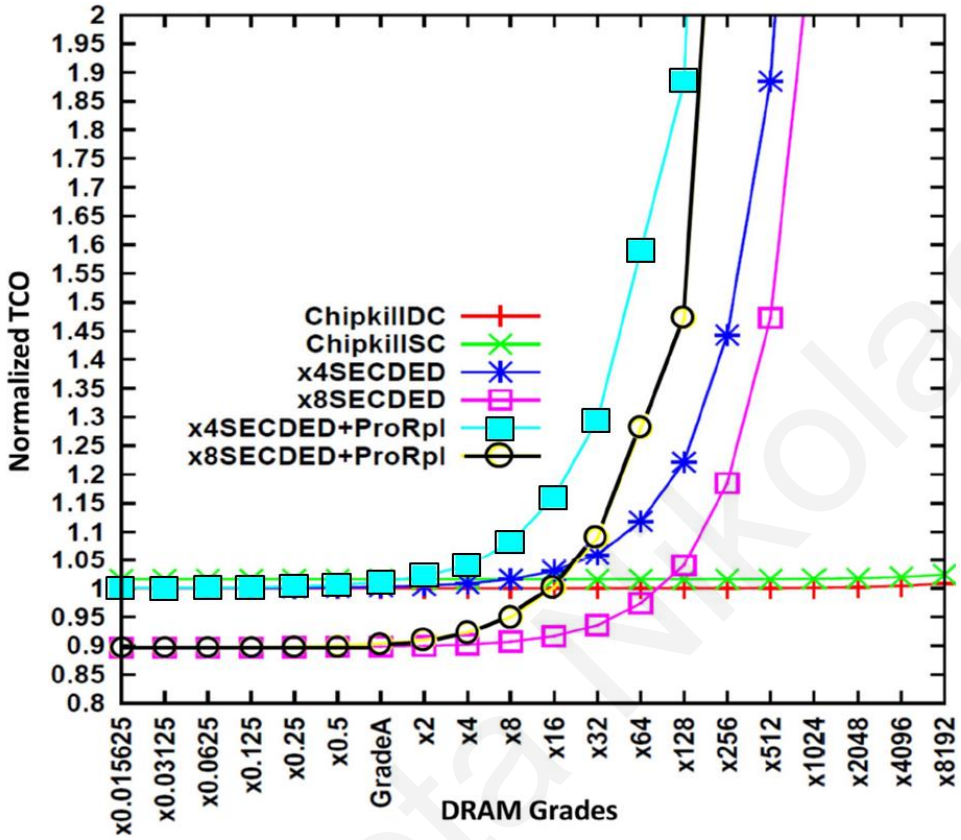

Figure 29: TCO results for different DRAM grades

For now, we consider DIMM cost to be the same for all the grades (we examine the cost issue later). All grades are correlated to GradeA by some factor (e.g. x2 is derived multiplying the fault rates of GradeA by two). The various curves correspond to different protection techniques and are normalized to the TCO of a DC using ChipkillDC with GradeA fault rates. Also, for both SECDED schemes we present results when the proactive replacement option is used whenever there is a correctable error.

As shown in Figure 29, the TCO of ChipkillDC and ChipkillSC are significantly less sensitive to increasing failure rates as compared to x4SECDED and x8SECDED.

Figure 29 also shows that the cost of the proactive replacement strategy is much higher than the savings it offers from reducing uncorrectable errors. Thus, there seems to be no opportunity to reduce the TCO by using the proactive replacement we evaluated.

Next, we explore the trade-off between DRAM cost and reliability. The results presented in Figure 30 shows what the DIMM cost for each protection technique should be to maintain the TCO of GradeA in all the other grades.

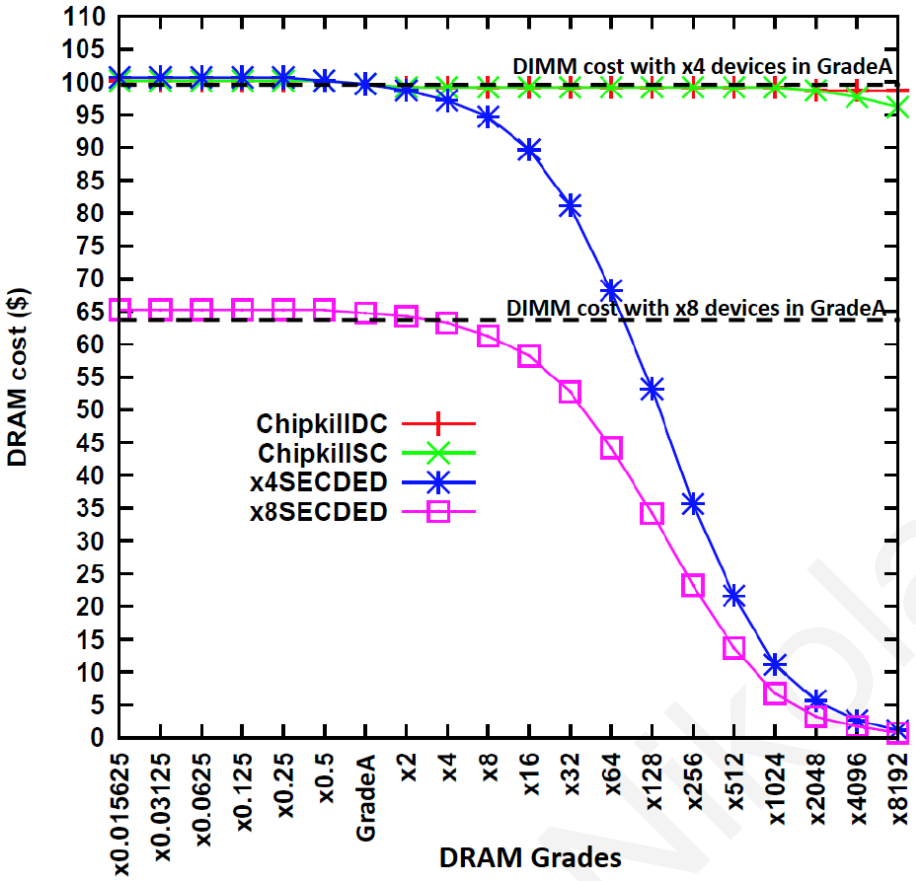

Figure 30: DRAM cost in ($) for different fault rates (DRAM grades)

Figure 30 effectively shows the opportunity to reduce the TCO for each protection technique, for all the grades, in the cases where the DIMM cost is below the corresponding curve for each protection technique. Choosing a DIMM cost below each curve leads to lower TCO than GradeA. For both SECDED schemes, a significant reduction of the DIMM cost is required to maintain the TCO of GradeA (for x4SECDED 70$ less per DIMM for a x256 grade). On the other hand, the results reveal that ChipkillDC (ChipkillSC) with a x4096 (x1024) grade can achieve the TCO cost of GradeA, with only 1$ cost reduction per DIMM.

6.4.3 ChipkillDC and ChipkillSC Performance, Energy and TCO for Online and Offline Jobs

To compare the performance of ChipkillDC and ChipkillSC, we run Online and Offline services in isolation and collocated on a server using both memory protection schemes. In Figure 31 we present the performance in terms of the Average Search Time (ms) and the 99th percentile Tail Latency (ms) of Web Search (WS) while running it alone and co-running it with different types of Offline services, on the same server.

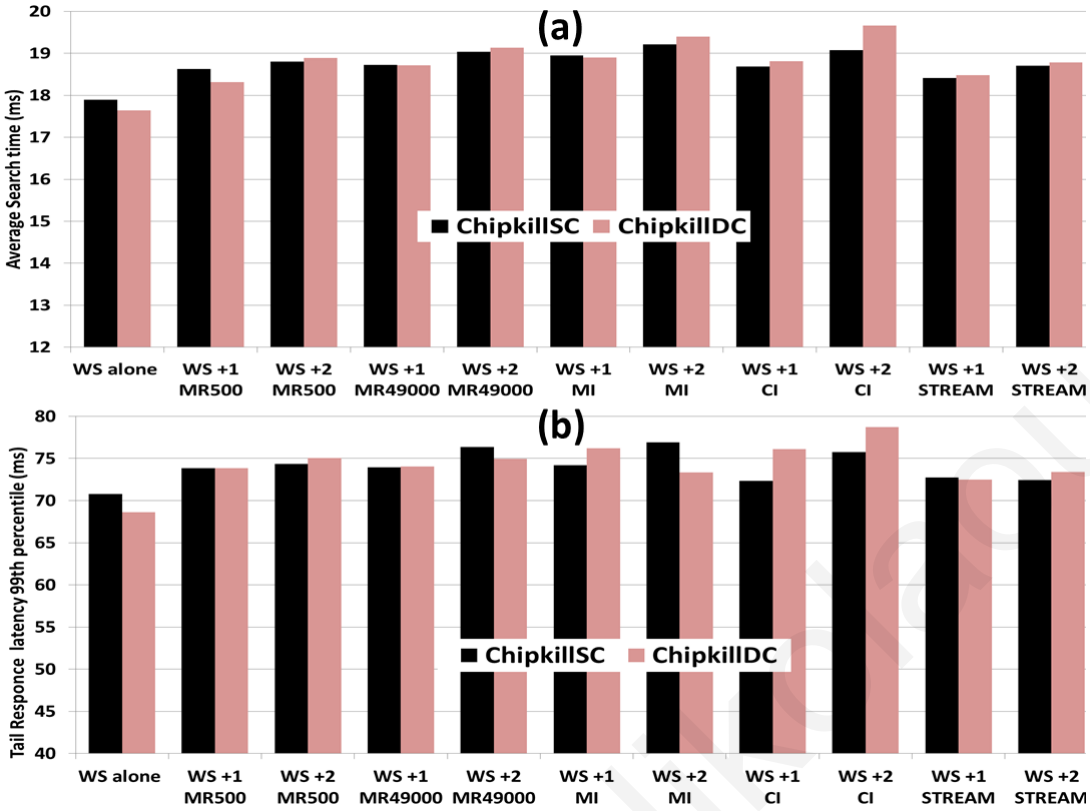

Figure 31: Performance overhead of ChipkillDC compared to ChipkillSC for different co-running configurations (a) performance in terms of Average Search Time and (b) performance measured in terms of 99% tail latency

The results are presented for different workload combinations. Specifically, two threads of Web search are running (two index servers) on two cores for all the configurations (recall that only two cores are used to meet the QoS constraint) while one or two instances of the same Offline service are co-running with Web Search. Each combination contains in total three or four threads, where each one is running on a separate core.

As we can observe from Figure 31.(a) and Figure 31.(b), when Web Search (WS) is co-running with Offline services, the Average Search Time and tail latency increase for both protection techniques as compared to Web Search running alone.

In Figure 31.(a) we observe that in most cases when not all cores are used, the performance is higher with ChipkillDC, the more bandwidth demanding protection technique, than ChipkillSC. We attribute this to the low memory pressure of Web Search and the ability of ChipkillDC to receive a codeword after a single burst which can enable faster forwarding of first word from memory. This reasoning appears consistent with the observation, in Figure 31.(a), that ChipkillDC is always worse when all cores are used (cases where 2 instances of Web Search run with 2 offline services) and the demand for memory bandwidth is higher.

Another important observation is that the trends presented in Figure 31.(a) are not the same with the trends in Figure 31.(b). For example, the average search time for ChipkillDC when co-running Web Search with two Memory Intensive services (seventh pair of bars in Figure 31.(a)) is higher than the average search time of ChipkillSC. On the other hand, ChipkillSC has higher 99th percentile tail response latency (seventh pair of bars in Figure 31.(b)) from ChipkillDC while using the same configuration (currently the TCO considers PD due to Average Search time. The framework can be extended to consider also the tail response latency in the TCO model and other performance metrics, as well).

Figure 32 presents the TCO of ChipkillDC and ChipkillSC (bars) when running two Web Search services (WS) on two cores per server and co-running with one or two instances of the Offline services.

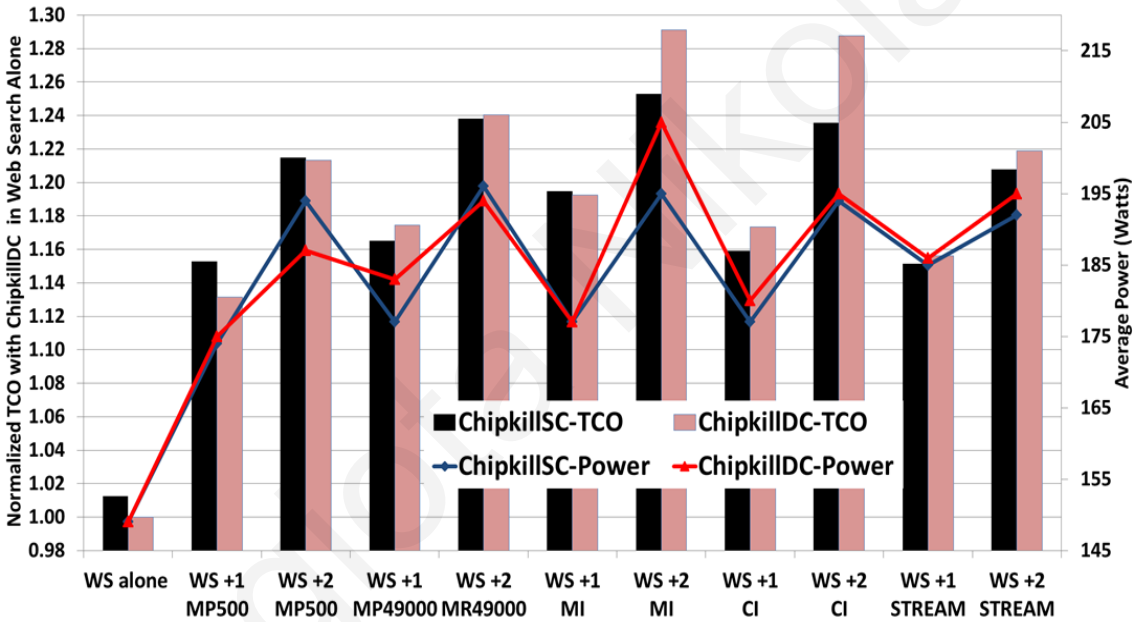

Figure 32: TCO results for collocated services considering Average Search Time and Average Power consumption

The results presented are normalized to the TCO of ChipkillDC when running Web Search service alone. An important observation from Figure 32 is that when two Web Search services are co-running with an Offline service, there is an increase in the TCO for both ChipkillDC and ChipkillSC compared to the TCO of Web Search running alone. This increase is due to the performance degradation and energy overheads, caused from the co-location of Web Search with Offline service. The power consumption for each protection technique is presented in Figure 32 with a secondary axis (lines). To have more accurate TCO results, we considered both performance and power parameters in the TCO. Considering only one of the two parameters can lead to inaccurate TCO results.

As seen in Figure 31(a), the performance degradation of Web Search when co-run with two instances of Map Reduce with 500MB dataset size (MR500) is larger for ChipkillDC. For the same scenario in Figure 32, the average power trends are reversed with ChipkillDC consuming less power than ChipkillSC. Considering both parameters in the TCO, Figure 32, shows that ChipkillDC has lower TCO than ChipkillSC.

6.4.4 Implications of NDEs on the System Reliability

Next, we analyze the system MTTF for SDC errors. As mentioned earlier, NDE errors can cause Silent Data Corruption (SDC). Determining the actual fraction of NDEs that lead to SDCs is a challenging problem [130]. A recent work, reports less than thousand SDCs from billions of queries performing fault injection in DRAM while running a Web Search application [30]. Figure 33 shows the effects of SDC errors in the system's MTTF SDC. We analyze the implications in the System MTTF SDC for a hypothetical range of SDC derating factors that derate the initial NDE rates, varying from 0 to 1. The x-axis represents the derating factor rates (NDE rates that cause SDCs), whereas the y-axis represents the system MTTF for SDC errors. The graph shows a number of curves each representing a different error protection scheme. The target reliability for SDC in this experiment is set to 3 years.
The results show that for ChipkillDC and ChipkillSC with SDC Derating factor=0.001 the system MTTF_SDC is 1016 and 1013 years respectively. For the largest derating factor, 1, the system MTTF_SDC if 1014 and 1010 years, respectively. However, with x8 and x4 SECDED, the system MTTF_SDC is many orders of magnitude lower as compared to Chipkill schemes, and with increasing derating factor it even becomes lower than the three-year MTTF target. Figure 33 also shows how the low average system utilization (31%) increases the MTTF SDC of all protection technique. Lower server utilization can provide higher reliability because it is more likely for errors to occur in idle DC resources. The above analysis indicates that the two Chipkill schemes provide much better resiliency against SDC errors which is important to consider when choosing DC DRAM memory protection.

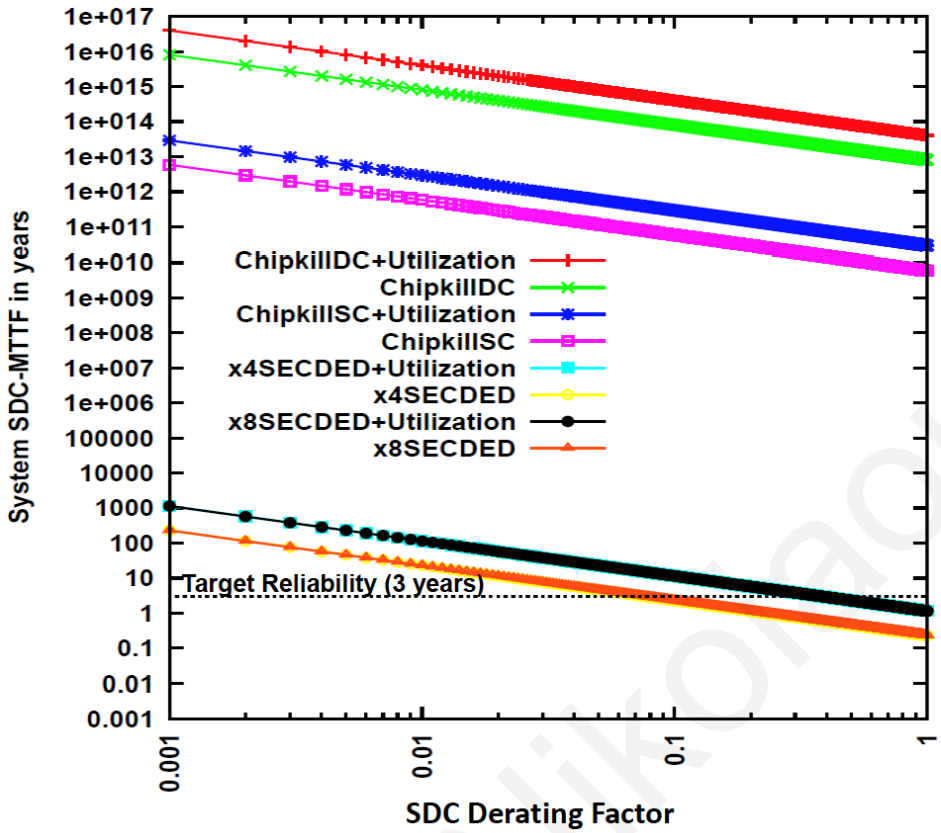

Figure 33: Design space exploration of NDEs that lead to SDCs for each protection technique + system utilization

6.4.5 TCO Sensitivity Analysis

Figure 34 shows a sensitivity analysis that aims to highlight the significance of considering some of the basic framework parameters when making TCO trade-offs related to memory protection in a DC. All the TCO numbers are presented in terms of dollars.

The first two bars show the total TCO for ChipkillSC and ChipkillDC when co-running 2 Web Search with 2 instances of the Memory Intensive (MCF) service and the framework includes all the parameters examined in this work. As the figure depicts the TCO savings that can be derived by including all the parameters are around 80000 dollars by selecting ChipkillSC. The second and third group of results present how TCO is affected by ignoring temperature and utilization. In these cases, we can observe a TCO increase for both protection techniques. Furthermore, the results show that ignoring co-location can have a large impact on TCO (lower TCO) and changes the result for which of the two protection techniques is better. Finally, excluding all the other parameters (performance, DRAM reliability and the contribution of operational costs to the TCO) reveals even larger sensitivity with both schemes seemingly having the same TCO.

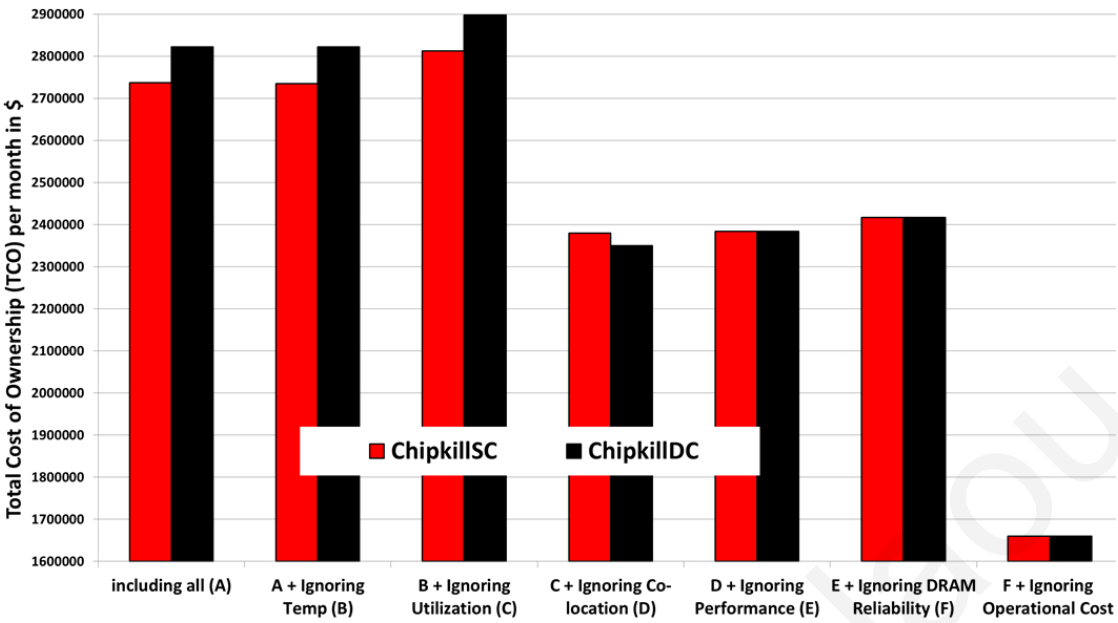

Figure 34: TCO sensitivity analysis of 2 Web Search + 2 Memory Intensive

6.5 AMPRA Model Validation and Insights

Overall, the findings of this analysis calls for manufacturers, vendor designers, datacenter owners and researchers to select/design memory protection schemes that maximize the TCO using the AMPRA proposed framework. The framework can also be useful to estimate the premium that compute as a service provider need to charge for running offline services to make up for the increase in TCO due to co-location. Another framework use is to enable processor designers to quantify the TCO benefits of new ECC options. Finally, this work offers a common framework for future research in this area that can be easily extended to explore new trade-offs. Moreover, datacenter designers can exploit these findings if they have the option to select lower cost and less reliable DRAM components. On the other side, a processor designer may benefit by performing a "what if" analysis to determine the potential TCO benefits of a new ECC code before embarking on building it.

AMPRA framework is useful for identifying key parameters that need to be considered when exploring different trade-offs. In the absence of such framework, some of these parameters may be overlooked. To validate AMPRA model we validate each individual model and components. Specifically, we validate power and temperature numbers using publicly available data, performance degradation running the applications on real hardware and we compare all the FIT rates with previous published papers.

Chapter 7

7 Conclusions and Future work

7.1 Conclusions

During the last few years, DCs have spread across the globe and they have increased in numbers, size and utilization. Large DCs that consist of thousands to tens of thousands of servers are used to deliver services, such as e-mail, web search, social networking, maps and face recognition, to billions of users. Additionally, a new paradigm has emerged that promotes the offering of services at the Edge, closer to users. One key challenge of these innovations is to limit their cost and energy consumption and, consequently, there is a growing need for efficient methodologies and techniques to optimize DC's design cost.

This work investigates a number of key parameters that affect TCO by evaluating DCs design decisions.

In this thesis we firstly, select the monitors and knobs to use to configure a computing system running an application while satisfying the application's requirements and not violating any system constraints. The selection relies on a heuristic correlation analysis between monitors and knobs to determine the minimum subset of monitors to observe and knobs to explore to determine the optimal system configuration for the application in order to provide optimization for the TCO of a system. At the end of this analysis, we reduce an 11-dimensional space to a 4-dimensional space for monitors and a 6-dimensional space to a 4-dimensional space for knobs. As presented in the thesis, TCO is mainly correlated with reliability, power and performance aspects. To optimize TCO, we provide different TCO evaluations that tackle all these aspects.

Secondly, we investigate the benefits of running an emerging security focused IoT application, (jamming detection), at the Edge vs. the Cloud by developing an end-to-end TCO model, which considers the application's requirements as well as the Edge's constraints. For the first time, we build such a model based on realistic performance and energy-efficiency measurements obtained from commodity 64-bit ARM based micro-servers that are excellent candidates for supporting Cloud services at the Edge. Such servers

represent the type of devices that can provide the right balance between power and performance, without requiring any complicate cooling and power supply infrastructure, which will not be available at the de-centralized deployments. Aiming at improving the energy efficiency, we exploit the pessimistic design margins adopted conventionally in such devices and investigate their operation under lower than nominal supply voltage and memory refresh-rate. Our results show that the jamming detection application deployed at an Edge environment is superior to a Cloud based solution by up to 2.13 times in terms of TCO. Moreover, when servers operate below nominal conditions, we can achieve up to 9% power savings which enables in several situations 100% gains in the TCO/area-coverage metric, i.e double area can be served with the same TCO.

Moreover, we present a comprehensive correlation analysis of an application's CPU-Vmin with hardware's performance counters on a real multicore system. The analysis reveals that a subset of the performance counters- the same ones across different workloads- have a strong correlation with a workload's CPU-Vmin. Moreover, the results show that the CPU-Vmin is accurately identifiable by monitoring a workload's performance-counters during the n-first seconds of its execution. Our findings serve as the basis of a software-based CPU-Vmin identification method that monitors an application for the first n-seconds and then sets the CPU supply voltage to a specific value for the rest of the execution. Our evaluation shows that the CPU-Vmin workload identification method provides a safe CPU-Vmin, 99.4% of the times, reduces power on average by 7.1% and provides substantial TCO savings when n-first equals to 20 seconds.

Finally, we propose for the first time, the AMPRA framework for modeling the implications of DRAM failures and DRAM error protection techniques on the TCO of a datacenter. The framework captures the effects and interactions of several key parameters including: the choice of DRAM protection technique (e.g. single vs dual channel Chipkill), device width (x4 or x8), memory size, power, FITs for various failure modes, the performance, power and temperature overheads of a protection technique for a given service and mixes of collocated services. The usefulness of the proposed framework is demonstrated through several case studies that identify the best DRAM protection technique in each case, in terms of TCO. Interestingly, our analysis reveals that among the three DRAM protection techniques considered, there is no one that is always superior to all the others. Moreover, each technique is better than the others for some cases. This underlines the importance and the need of the proposed framework for making optimal memory protection datacenter design decisions. As part of this work, we analyze and report the performance and power with single channel and dual channel Chipkill on real hardware when running a web search benchmark alone and

collocated with benchmarks of varying memory intensity. This analysis reveals that the choice of memory protection can have serious performance and TCO ramifications depending on the memory characteristics of collocated services. Other analysis reveals that, for the datacenter and services assumed in this study, when using Chipkill protection it can be beneficial for TCO to use DRAM with 100x the failure rate of a baseline DRAM as long as the cost per DIMM is at least a dollar less compared to the baseline.

7.2 Lessons Learnt

This Section recapitulates the thesis impact and at the same time it serves as a springboard to future research and development- based on the insights gained during the thesis process.
Each of the contributions of this thesis has had a significant impact on both industry and research community. Particularly, AMPRA tool[2] has been placed online (more than 1000 downloads by now) and organizations expressed interest in this tool and had the opportunity to use it in their analyses.
Moreover, during my two months internship at IROC technologies, I encapsulated AMPRA tool on the development of a reliability-based architect tool. This tool uses AMPRA to estimate the TCO of a new developed system, including reliability analysis of all the components which are part of this system.
The findings of this thesis pave the way of new future directions. However, some of these directions have negative preliminary results. The dynamic reconfiguration of the system according to the TCO savings is one of them. Specifically, this idea has been based on the dynamic change of different system knobs such as memory protection, memory interleaving and turbo mode, according to the application needs in order to optimize the TCO. Nevertheless, this idea was not as successful as it had been initially expected since the default settings for the specific applications served as the best choice in terms of TCO.

7.3 Future Work

In spite of the technical contributions of the current text, it is clear that improvements and optimizations can improve the technology of the concepts discussed herein. We propose future work items, corresponding to each technical Chapter of the current text:

[2] (Publicly Available- AMPRA: Analyzer of TCO Implications of Memory Failures and Memory Protection http://www2.cs.ucy.ac.cy/carch/xi/ampra tco.php)

Chapter 3: A possible direction to this work is to explore other configurations, monitors and knobs and investigate the generality of the observations to other platforms and applications.

Chapter 4: This work provides a clear motivation for even more power-efficient solutions at the Edge and the use of such evaluation metrics. An important future direction is the inclusion of the sensors in the evaluation in order to include the aspect of battery life and many other parameters consisting the embedded systems. Moreover, distributed Cloud is widely used by many clients. However, these clients are kept agnostic of the location that they run their application and the security issues that may face. The cautious choice of the location dependent on the TCO will be very beneficial. Another important future direction would be to extend the research of this work by studying more applications.

Chapter 5: The identification method highlights the need of the development of a software DVS governor that predicts the workloads CPU-Vmin from the first n-seconds. Another important direction for future work is the further reduction of false positives without increasing false negatives and thus improve power savings. Furthermore, the methodology can be used to predict viruses such as the di/dt virus in platforms that do not have droop counters visibility such as X-Gene2 platform. Finally, future work should investigate the generality of the observations made on X-Gene 2 for other platforms.

Chapter 6: TCO evaluation of DRAM protections is another domain that invites meaningful future contributions. One possible direction is to explore the TCO of other components (e.g. SSDs), study the cost-benefits of new ECC schemes and other DRAM technologies.

Bibliography

[1] Tim Reeve and Barb Everdene, “Applying Total Cost of Ownership to Sustainability Purchasing”, Workbook (Version 1.0) on Sustainability Purchasing Network

[2] Amazon Web Services, “AWS Total Cost of Ownership (TCO) Calculator”, 2019

[3] Microsoft Azure, “Total Cost of Ownership (TCO) Calculator”, 2019

[4] L. A. Barroso, J. Clidaras, and U. Holzle, “The datacenter as a computer: An introduction to the design of warehouse-scale machines,” Synthesis lectures on Computer Architecture, pp. 1–154, 2013.

[5] E. Brewer, “Lessons from giant-scale services,” Internet Computing, pp. 46–55, 2001.

[6] Bonomi, Flavio, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. "Fog computing and its role in the internet of things." In Proceedings of the first edition of the MCC workshop on Mobile Cloud computing, pp. 13-16. ACM, 2012

[7] Shekhar, Shashank, Ajay Dev Chhokra, Anirban Bhattacharjee, Guillaume Aupy, and Aniruddha Gokhale. "INDICES: exploiting Edge resources for performance-aware Cloud-hosted services." In Fog and Edge Computing (ICFEC), 2017 IEEE 1st International Conference on, pp. 75- 80. IEEE, 2017.

[8] Tong, Liang, Yong Li, and Wei Gao. "A hierarchical Edge Cloud architecture for mobile computing." INFOCOM 2016, The 35th Annual IEEE International Conference on Computer Communications, IEEE. IEEE, 2016.

[9] El-Sayed, Hesham, Sharmi Sankar, Mukesh Prasad, Deepak Puthal, Akshansh Gupta, Manoranjan Mohanty, and ChinTeng Lin. "Edge of things: the big picture on the integration of Edge, IoT and the Cloud in a distributed computing environment." IEEE Access 6 (2018): 1706-1717.

[10] C. Lefurgy, X. Wang, and M. Ware, “Power capping: A prelude to power shifting,” Cluster Computing, vol. 11, no. 2, 2008.

[11] Luiz Andre Barroso and Urs Holzle, The Datacenter as a Computer. Morgan Claypool, 2009.

[12] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, “Ensemble level power management for dense blade servers,” in Proceedings of the 33rd Annual International Symposium on Computer Architecture (ISCA), pp. 66-77, 2006.

[13] X. Wang, M. Chen, C. Lefurgy, and T. W. Keller, “SHIP: Scalable hierarchical power control for large-scale data centers”, in Proceedings of the 18th International Conference on Parallel Architectures and Compilation Techniques (PACT), pp. 91-100, 2009.

[14] R. Joseph, D. Brooks, and M. Martonosi. “Control techniques to eliminate voltage emergencies in high performance processors”. In Proc. International Symposium on High-Performance Computer Architecture, pp.79-90, 2003

[15] Bertran, Ramon, Alper Buyuktosunoglu, Pradip Bose, Timothy J. Slegel, Gerard Salem, Sean Carey, Richard F. Rizzolo, and Thomas Strach. "Voltage noise in multi-core processors: Empirical characterization and optimization opportunities." In Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on, pp. 368-380. IEEE, 2014.

[16] Reddi, Vijay Janapa, Svilen Kanev, Wonyoung Kim, Simone Campanoni, Michael D. Smith, Gu-Yeon Wei, and David Brooks. "Voltage noise in production processors." IEEE micro 31, no. 1 (2011): 20-28.

[17] Kim, Youngtaek, Lizy Kurian John, Sanjay Pant, Srilatha Manne, Michael Schulte, William Lloyd Bircher, and Madhu Saravana Sibi Govindan. "AUDIT: Stress testing the automatic way." In Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on, pp. 212-223. IEEE, 2012.

[18] Whatmough, Paul N., Shidhartha Das, Zacharias Hadjilambrou, and David M. Bull. "Power Integrity Analysis of a 28 nm Dual-Core ARM Cortex-A57 Cluster Using an All-Digital Power Delivery Monitor." IEEE Journal of Solid-State Circuits 52, no. 6 (2017): 1643-1654.

[19] M. S. Gupta, K. Rangan, M. D. Smith, G.-Y. Wei, and D. M. Brooks. DeCoR: A Delayed Commit and Rollback Mechanism for Handling Inductive Noise in Processors. In HPCA ’08, 2008.

[20] Abbas, Muhamed Fauzi Bin, et al. "Hardware performance counters based runtime anomaly detection using SVM." 2017 TRON Symposium (TRONSHOW). IEEE, 2017.

[21] Eyerman, Stijn, et al. "A performance counter architecture for computing accurate CPI components." ACM SIGOPS Operating Systems Review 40.5 (2006): 175-184.

[22] Reddi, Vijay Janapa, et al. " Voltage emergency prediction: Using signatures to reduce operating margins." 2009 IEEE 15th International Symposium on High Performance Computer Architecture. IEEE, 2009.

[23] Reddi, Vijay Janapa, et al. "Voltage noise: Why it's bad, and what to do about it." 5th IEEE Workshop on Silicon Errors in Logic-System Effects (SELSE), Palo Alto, CA. 2009.

[24] Hennessy, John L. Hennessy and Patterson, David A. Computer Architecture, Fourth Edition: "A Quantitative Approach". Morgan Kaufmann. San Francisco, CA, USA : Publishers Inc., 2006.

[25] Naveen Muralimanohar, Rajeev Balasubramonian, Norman P. Jouppi . "CACTI 6.0: A Tool to Model Large Caches". s.l. : HP, 2009.

[26] Micron. [Online] http://www.micron.com/products/support/power-calc.

[27] HP. HP iLO 3 User Guide. [Online] 2014. http://h20628.www2.hp.com/kmext/ kmcsdirect/emr_na- c02774507- 6.pdf.

[28] Intel. "likwid-powermeter: Tool for accessing RAPL counters on Intel processor." [Online] https://github.com/RRZE-HPC/ likwid/wiki/Likwid-Powermeter.

[29] B. Grot et al. "Optimizing Data-Center TCO with Scale-Out Processors". 2008. IEEE Micro

[30] Y. Luo, S. Govindan, B. Sharma, M. Santaniello, J. Meza, A. Kansal, J. Liu, B. Khessib, K. Vaid, and O. Mutlu "Characterizing application memory error vulnerability to optimize datacenter cost via heterogeneous-reliability memory", 44th Annual International Conference on Dependable Systems and Networks, pp. 467-478, 2014.

[31] Xia, Feng, Laurence T. Yang, Lizhe Wang, and Alexey Vinel. "Internet of things." International Journal of Communication Systems 25, no. 9 (2012).

[32] Cisco Data in Motion (DMo) technology allows data management and analysis of large volumes of data coming through IoT at the edge

[33] W. Shi, G. Pallis and Z. Xu, "Edge Computing [Scanning the Issue]," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1474-1481, Aug. 2019.

[34] Tan, Lu, and Neng Wang. "Future internet: The internet of things." In 2010 3rd international conference on advanced computer theory and engineering (ICACTE), vol. 5, pp. V5-376. IEEE, 2010.

[35] Mao, Yuyi, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. "A survey on mobile edge computing: The communication perspective." IEEE Communications Surveys & Tutorials 19, no. 4 (2017): 2322-2358.

[36] Stojmenovic, Ivan. "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks." In 2014 Australasian Telecommunication Networks and Applications Conference (ATNAC), pp. 117-122. IEEE, 2014.

[37] Bahtovski, Aleksandar, and Marjan Gusev. "Cloudlet Challenges." Procedia Engineering 69 (2014): 704-711.

[38] Satyanarayanan, M, Bahl P., Caceres R., and Davies N. "The case for vm-based cloudlets in mobile computing." IEEE Pervasive Computing 8, no. 4 (2009): 14-23.

[39] Bilal, Kashif, Osman Khalid, Aiman Erbad, and Samee U. Khan. "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers." Computer Networks 130 (2018): 94-120.

[40] R.C.Baumann “Soft errors in advanced semiconductor devices-part1”: The three radiation sources.. Mar.2001. IEEE Transaction on Device and Materials Reliability.

[41] V. Sridharan, J. Stearley, N. DeBardeleben, S. Blanchard, and S. Gurumurthi, “Feng shui of supercomputer memory: Positional effects in dram and sram faults”. 2013. International Conference on High Performance Computing, Networking, Storage and Analysis. pp. 22:1–22:11.

[42] V. Sridharan and D. Liberty, “A study of dram failures in the field.” 2012. International Conference on High Performance Computing, Networking, Storage and Analysis. pp. 76:1– 76:11.

[43] Stefanos Kaxiras and Margaret Martonosi. “Computer Architecture Techniques for Power-Efficiency “(1st ed.). s.l. : Morgan and Claypool , 2008.

[44] Jacob, Bruce, Ng, Spencer and Wang, David. “Memory Systems: Cache, Dram, Disk”.Morgan Kaufmann. San Francisco, CA, USA : Inc, 2007.

[45] R. W. Hamming, “Error detecting and error correcting codes”. 1950. Bell System Technical Journal. pp. 147–160.

[46] M. Y. Hsiao, “A class of optimal minimum odd-weight-column sec-ded codes”. 1970. BM J. Res. Dev. pp. 395–401.

[47] Timothy J Dell, “A white paper on the benefits of chipkill-correct ecc for pc server main memory”.1997. IBM Microelectronics Division. pp. 1-23

[48] A. Portero, S. Kuchaˇr, R. Vavˇr´ık, M. Golasowski, and V. Vondr´a. “System and application scenarios for disaster management processes, the rainfall-runoff model case study”. In IFIP International Conference on Computer Information Systems and Industrial Management, pages 315–326. Springer, 2014.

[49] WorldSensing, https://www.worldsensing.com

[50] “Cloudsuite web search site,” http://parsa.epfl.ch/cloudsuite/search.html.

[51] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, “Clearing the clouds: A study of emerging scale-out workloads on modern hardware,” in Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 37–48, 2012.

[52] “Standard performance evaluation corporation. spec cpu 2006,” 2006. http://www.spec.org/cpu2006/.

[53] “Standard performance evaluation corporation. spec cpu 2017,” 2017. https://www.spec.org/cpu2017/.

[54] NAS Parallel Benchmarks Suite, v3.3.1. https://www.nas.nasa.gov/publications/npb.html#url

[55] A. Iordache, E. Buyukkaya, and G. Pierre. “Heterogeneous resource selection for arbitrary hpc applications in the cloud”, in IFIP International Conference on Distributed Applications and Interoperable Systems, pages 108–123. Springer, 2015.

[56] K. Hoste and L. Eeckhout. “Comparing benchmarks using key microarchitecture-independent characteristics”. In 2006 IEEE International Symposium on Workload Characterization, pages 83–92. IEEE, 2006.

[57] K. Hoste and L. Eeckhout. “Microarchitecture-independent workload characterization”. IEEE Micro, 27(3):63–72, 2007.

[58] K. Hoste, A. Phansalkar, L. Eeckhout, A. Georges, L. K. John, and K. De Bosschere. “Performance prediction based on inherent program similarity”, In Proceedings of the 15th international conference on Parallel architectures and compilation techniques, pages 114–122. ACM, 2006.

[59] M. Annavaram, R. Rakvic, M. Polito, J.-Y. Bouguet, R. A. Hankins, and B. Davies. “The fuzzy correlation between code and performance predictability”, In

Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture, pages 93–104. IEEE Computer Society, 2004.

[60] H. Abdi and L. J. Williams. “Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics,” 2(4):433–459, 2010.

[61] I. Jolliffe. “Principal component analysis”. Wiley Online Library, 2002.

[62] S. Kuchar, M. Golasowski, R. Vavrik, M. Podhoranyi, B. Sir, and J. Martinovic. “Using high performance computing for online flood monitoring and prediction”. International Journal of Environmental, Ecological, Geological and Geophysical Engineering, 9(5):267–272, 2015.

[63] P. Nikolaou, Y. Sazeides, L. Ndreu, and M. Kleanthous. “Modeling the implications of dram failures and protection techniques on datacenter tco”. In Proceedings of the 48th International Symposium on Microarchitecture, MICRO-48, pages 572–584, New York, NY, USA, 2015. ACM.

[64] J. Mars, L. Tang, R. Hundt, K. Skadron, and M. L. Soffa. “Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations”. In 44th Annual International Symposium on Microarchitecture, pages 248–259, 2011.

[65] H. Yang, A. Breslow, J. Mars, and L. Tang. “Bubble-flux: Precise online qos management for increased utilization in warehouse scale computers”. In 40th Annual International Symposium on Computer Architecture, pages 607–618, 2013.

[66] A. Portero, R. Vavrik, S. Kuchar, M. Golasowski, V. Vondrak, S. Libutti, G. Massari, W. Fornaciari, and I. e Bioengegneria. “Flood prediction model simulation with heterogeneous trade-offs in high performance computing framework”. In 29th EUROPEAN Conference on Modelling and Simulation ECMS, 2015.

[67] D. M. Tullsen, S. J. Eggers, and H. M. Levy. “Simultaneous multithreading: Maximizing on-chip parallelism”. SIGARCH Comput. Archit. News, 23(2):392–403, May 1995.

[68] M. Weiser, B. Welch, A. Demers, and S. Shenker. “Scheduling for reduced cpu energy”. In Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation, OSDI ’94, Berkeley, CA, USA, 1994. USENIX Association.

[69] J. M. Tendler, J. S. Dodson, J. Fields, H. Le, and B. Sinharoy. “Power4 system microarchitecture”. IBM Journal of Research and Development, 46(1):5–25, 2002.

[70] Intel. “Intel Turbo Boost Technology in Intel Core microarchitecture (Nehalem) based processors”. White paper. Technical report, November 2008.

[71] L. Huang and Q. Xu. “Characterizing the lifetime reliability of manycore processors with core-level redundancy”. In 2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 680–685. IEEE, 2010.

[72] “BIOS and Kernel Developers Guide (BKDG) for AMD Family 10h”. April 2010.

[73] S. Ankireddi and T. Chen. “Configuring and using DDR3 memory with HP ProLiant Gen8 Servers”, Best Practice Guidelines for ProLiant servers with Intel Xeon processors. February 2014.

[74] “BIOS and Kernel Developers Guide (BKDG) for AMD Family 15h”. February 2014.

[75] J. H. Ahn, N. P. Jouppi, C. Kozyrakis, J. Leverich, and R. S. Schreiber. “Future scaling of processor-memory interfaces”. In Conference on High Performance Computing Networking, Storage and Analysis, pages 42:1–42:12, 2009.

[76] X. Jian, H. Duwe, J. Sartori, V. Sridharan, and R. Kumar. “Low-power, low-storage-overhead chipkill correct via multi-line error correction”. In International Conference on High Performance Computing, Networking, Storage and Analysis, pages 24:1–24:12, 2013.

[77] A. N. Udipi, N. Muralimanohar, R. Balsubramonian, A. Davis, and N. P. Jouppi. “Lot-ECC: Localized and tiered reliability mechanisms for commodity memory systems”. In 39th Annual International Symposium on Computer Architecture, pages 285–296, 2012.

[78] Hp rom-based setup utility user guide. February HP TR, 2014.

[79] lm-sensors. http://www.lm-sensors.org/wiki/man/sensors-detect.

[80] D. Hardy, M. Kleanthous, I. Sideris, A. Saidi, E. Ozer, and Y. Sazeides. “An analytical framework for estimating tco and exploring data center design space”. In International Symposium on Performance Analysis of Systems and Software, pages 54–63, 2013.

[81] R. C. Team et al. “R: A language and environment for statistical computing”. 2013.

[82] J. Hamilton, “Architecture for modular data centers,” arXiv preprint cs/0612110, 2006.

[83] B. Schroeder and G. A. Gibson, “Understanding failures in petascale computers,” Journal of Physics: Conference Series, 2007, Vol. 78, No 1, p.012022.

[84] J. Daly, B. Harrod, T. Hoang, L. Nowell, B. Adolf, S. Borkar, N. DeBardeleben, M. Heroux, D. Rogers, R. R. ANL, et al., “Inter-Agency Workshop on HPC Resilience at Extreme Scale,” 2012.

[85] B. Schroeder and G. Gibson, “A large-scale study of failures in high-performance computing systems,” Transactions on Dependable and Secure Computing, pp. 337–350, 2010.

[86] B. Schroeder, E. Pinheiro, and D. Weber, “Dram errors in the wild: a large-scale field study,” SIGMETRICS, pp. 193–204, June 2009.

[87] L. Borucki, G. Schindlbeck, and C. Slayman, “Comparison of accelerated dram soft error rates measured at component and system level,” in International Reliability Physics Symposium, pp. 482–487, April 2008.

[88] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. Wenisch, “Disaggregated memory for expansion and sharing in blade servers,” in 36th Annual International Symposium on Computer Architecture, pp. 267–278, 2009.

[89] G. Daniel Bowers, “Server trends,” TR, 2012.

[90] C. Constantinescu, “Impact of deep submicron technology on dependability of vlsi circuits,” in International Conference on Dependable Systems and Networks, pp. 205–209, 2002.

[91] C. Weaver, J. Emer, S. Mukherjee, and S. Reinhardt, “Techniques to reduce the soft error rate of a high-performance microprocessor,” in 31st Annual International Symposium on Computer Architecture, pp. 264–275, June 2004.

[92] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, “A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor,” in 36th Annual International Symposium on Microarchitecture, 2003.

[93] S. Ankireddi and T. Chen, “Challenges in thermal management of memory modules,” Electronics Cooling, February 2008.

[94] “Micron,2gb: x4, x8, x16 ddr3 sdram,” Datasheed:https://www.micron.com/products/datasheets.

[95] “Intel, Xeon Processor E7 Family:Reliability, Availability,and Serviceability, Advanced data integrity and resiliency support for mission-critical deployments,” June 2006.

[96] A. Kleen, “mcelog: memory error handling in user space, linux,” TR, 2010.

[97] D. Tang, P. Carruthers, Z. Totari, and M. Shapiro, “Assessment of the effect of memory page retirement on system ras against hardware faults,” in International Conference on Dependable Systems and Networks, pp. 365–370, 2006.

[98] D. Meisner, C. M. Sadler, L. A. Barroso, W.-D. Weber, and T. F. Wenisch, “Power management of online data-intensive services,” in 38th Annual International Symposium on Computer Architecture, pp. 319–330, 2011.

[99] L. A. Barroso, J. Dean, and U. H¨olzle, “Web search for a planet: The google cluster architecture,” IEEE Micro, pp. 22–28, Mar. 2003.

[100] C. Patel and A. Shah, “Cost model for planning, development and operation of a data center,” HP TR, 2005.

[101] J. Karidis, J. E. Moreira, and J. Moreno, “True value: assessing and optimizing the cost of computing at the data center level,” in 6th ACM Conference on Computing Frontiers, pp. 185–192, 2009.

[102] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, “Making scheduling ”cool”: temperature-aware workload placement in data centers,” in Annual Conference on USENIX, pp. 5–5, 2005.

[103] J. Koomey, K. Brill, P. Turner, J. Stanley, and B. Taylor, “A simple model for determining true total cost of ownership for data centers,” White Paper, Uptime Institute, 2007.

[104] K. V. Vishwanath, A. Greenberg, and D. A. Reed, “Modular data centers: how to design them?,” in 1st Workshop on Large-Scale System and Application Performance, pp. 3–10, 2009.

[105] S. Li, K. Chen, M.-Y. Hsieh, N. Muralimanohar, C. D. Kersey, J. B. Brockman, A. F. Rodrigues, and N. P. Jouppi, “System implications of memory reliability in exascale computing,” in International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 46:1–46:12, 2011.

[106] D. H. Yoon and M. Erez, "Virtualized and flexible ecc for main memory," in 15th Edition of Architectural Support for Programming Languages and Operating Systems, pp. 397–408, 2010.

[107] "Memory technology evolution: an overview of system memory technologies," December HP, TR, 2010.

[108] M. Guevara, B. Lubin, and B. C. Lee, "Market mechanisms for managing datacenters with heterogeneous microarchitectures," ACM Trans. Comput. Syst., pp. 3:1–3:31, Feb. 2014.

[109] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, "3d-ice: Fast compact transient thermal modeling for 3d ics with inter-tier liquid cooling," in International Conference on Computer-Aided Design, pp. 463–470, Nov 2010.

[110] I. Cecil Ho, CST, "Innovative testing puts fallout dram back into systems," January 2003. Simmtester.com.

[111] "Memory Test Background," 2000. http://tinyurl.com/m7c3wf7.

[112] Z. Al-Ars, "Dram fault analysis and test generation," Ph.D. dissertation, Delft, 2005.

[113] "Arrhenius equation," https://en.wikipedia.org/wiki/Arrhenius_equation.

[114] "Intel Xeon Processor E5620-cost," http://www.cpubenchmark.net/cpu.php?cpu=Intel+Xeon+ E5620+%40+2.40GHz.

[115] "Desktop Drive 500GB-cost and power," http://www.ebuyer.com/394432-wd-500gb-blackdesktop-drive-wd5003azex.

[116] "Intel Server Motherboard cost," http://www.cpusolutions.com/store/pc/IntelS1200V3RPS-Server-Motherboard-Intel-C222-ChipsetSocket/-H3-LGA-1150-p3673.htm#.U4OhLHZqOPM.

[117] "Intel cpu configuration," http://www.rect.coretoeurope.com/rack-server/1u-intel-server/2428-short1u-intel-single-cpu-rack-server.html.

[118] "Server case and power supply," http://www.newegg.com/ Product/Product.aspx?Item=N82E16811108235.

[119] "Kingston Technology ValueRAM 8GB-x4 1600MHz DDR3-cost," http://www.amazon.com/KingstonTechnology-PC3-12800-KVR16LR11S4-8HA/dp/B00BYO7CZM.

[120] "Kingston ValueRam 8GB-x8 1600 MHz DDR3-cost," http://www.amazon.com/Kingston-TechnologyValidated-KVR16LR11D8-8I/dp/B00JWFMBIS.

[121] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and qos-aware cluster management," in 19th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 127–144, 2014.

[122] "The International Technology Roadmap for Semiconductors, ITRS, Tech. Rep.," 2013. http://www.itrs.ne.

[123] "Hp advanced memory error detection technology," July TR, 2011.

[124] "Hp proliant dl380 g7 server user guide, 2nd edition," February 2011.

[125] "Prime95," http://www.mersenne.org/download/.

[126] J. Srinivasan, S. V. Adve, P. Bose, S. V. A. P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," in 31st International Symposium on Computer Architecture, pp. 276–287, 2004.

[127] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, and J. Hiller, "Exascale computing study: Technology challenges in achieving exascale systems," TR, 2008.

[128] "Intel Xeon Processor E3-power," http://www.servethehome.com/intel-xeon-e3-1220-v3benchmark-review-haswell-xeon/.

[129] J. Hamilton, "Overall data center costs." http://perspectives.mvdirona.com/2010/09/18/ OverallDataCenterCosts.aspxn.

[130] S. K. S. Hari, S. V. Adve, H. Naeimi, and P. Ramachandran, "Relyzer: Exploiting application-level fault equivalence to analyze application resiliency to transient faults," in Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 123–134, 2012.

[131] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013–2018. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-indexvni/white_paper_c11-520862.pdf

[132] Bonomi F. "Connected vehicles, the internet of things, and fog computing." The Eighth ACM Int. Workshop on VehiculAr InterNETworking VANET, Las Vegas, USA, 2011.

[133] Roman, Rodrigo, Javier Lopez, and Masahiro Mambo. "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges." Future Generation Computer Systems 78 (2018): 680-698.

[134] FCC public notice 202/418-0500, January 27, 2015 Enforcement Advisory No. 2015-01

[135] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, “Gaming in the Clouds: Qoe and the users perspective,” Mathematical and Computer Modelling, vol. 57, no. 11, pp. 2883–2894, 2013.

[136] R. Kohavi, R. M. Henne, and D. Sommerfield, “Practical guide to controlled experiments on the web: listen to your customers not to the hippo,” in 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007, pp. 959–967.

[137] Y. A. Wang, C. Huang, J. Li, and K. W. Ross, “Estimating the performance of hypothetical Cloud service deployments: A measurement-based approach,” in INFOCOM, 2011 Proceedings IEEE. IEEE, 2011, pp. 2372–2380.

[138] Luan, Tom H., Longxiang Gao, Zhi Li, Yang Xiang, Guiyi We, and Limin Sun. "A view of fog computing from networking perspective." ArXivPrepr. ArXiv160201509 (2016).

[139] V. Bahl, “Cloud 2020: Emergence of micro data centers (Cloudlets) for latency sensitive computing (keynote),” Middleware 2015, 2015.

[140] Andrew Froehlich, “How Edge Computing Compares with Cloud Computing”, Networking Computing Blog, 2018

[141] Chang, Yu-Shuo, and Shih-Hao Hung. "Developing Collaborative Applications with Mobile Cloud-A Case Study of Speech Recognition." J. Internet Serv. Inf. Secur. 1.1 (2011): 18-3

[142] Chang, Hyunseok, et al. "Bringing the cloud to the edge." Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on. IEEE, 2014..

[143] Clinch, Sarah, Jan Harkes, Adrian Friday, Nigel Davies, and Mahadev Satyanarayanan. "How close is close enough? Understanding the role of Cloudlets in supporting display appropriation by mobile users." In Pervasive Computing and

Communications (PerCom), 2012 IEEE International Conference on, pp. 122-127. IEEE, 2012.

[144] Fesehaye, Debessay, Yunlong Gao, Klara Nahrstedt, and Guijun Wang. "Impact of Cloudlets on interactive mobile Cloud applications." In Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International, pp. 123-132. IEEE, 2012.

[145] Shekhar, Shashank, and Aniruddha Gokhale. "Dynamic resource management across Cloud-Edge resources for performance-sensitive applications." In Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 707-710. IEEE Press, 2017.

[146] Powers, Nathaniel, Alexander Alling, Kiara Osolinsky, Tolga Soyata, Meng Zhu, Haoliang Wang, He Ba, Wendi Heinzelman, Jiye Shi, and Minseok Kwon. "The Cloudlet accelerator: Bringing mobile-Cloud face recognition into real-time." In Globecom Workshops (GC Wkshps), 2015 IEEE, pp. 1-7. IEEE, 2015.

[147] Soyata, Tolga, Rajani Muraleedharan, Colin Funai, Minseok Kwon, and Wendi Heinzelman. "Cloud-vision: Real-time face recognition using a mobile-Cloudlet-Cloud acceleration architecture." In Computers and communications (ISCC), 2012 IEEE symposium on, pp. 000059-000066. IEEE, 2012.

[148] http://road.cc/content/news/76427-thousands-using-gps-jammers-disguise-over-long-hours-or-stolen-cars

[149] https://www.cnet.com/news/truck-driver-has-gps-jammer-accidentally-jams-newark-airport/

[150] https://www.pdaelectronics.com/index.php

[151] http://www.suresafe.com.tw/Signal-Jammer-Detector.html

[152] https://simplisafe.com/home-3

[153] https://www.forbes.com/sites/marcwebertobias/2015/01/29/this-popular-wireless-alarm-system-can-be-hacked-with-a-magnet-and-scotch-tape/#6b466ec450de

[154] http://www.chronos.co.uk/en/news-and-pr/news-page/1216-chronos-enables-exelis-to-develop-new-capability-in-gps-interference-detection-and-geolocation

[155] http://www.chronos.co.uk/files/pdfs/pres/2010/GPSJamming/Generic_Briefing_Presentation.pdf

[156] https://www.gps.gov/cgsic/meetings/2014/curry.pdf

[157] http://www.sekotech.com/gsm-3g-4g-jammer-detector/st-171-gsm-3g-4g-gps-jammer-detector.html

[158] http://www.pki-electronic.com/products/jamming-systems/jammer-detector/

[159] https://play.google.com/store/apps/details?id=com.microcadsystems.serge.jammerdetector&hl=en

[160] Pelechrinis, Konstantinos, Marios Iliofotou, and Srikanth V. Krishnamurthy. "Denial of service attacks in wireless networks: The case of jammers." IEEE Communications surveys & tutorials 13.2 (2011): 245-257.

[161] Xu, Wenyuan, et al. "The feasibility of launching and detecting jamming attacks in wireless networks." Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing. ACM, 2005.

[162] "Intel cpu configuration," http://www.rect.coreto- europe.com/rack- server/1u-intel- server/2428- short- 1u- intel- single- cpu- rack- server.html.

[163] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, D. Gizopoulos, G. Favor, K. Sankaran and S. Das, "A System-Level Voltage/Frequency Scaling Characterization Framework for Multicore CPUs", IEEE Silicon Errors in Logic – System Effects (SELSE 2017), Boston, MA, USA, March 2017.

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

[165] G. Karakonstantis, K. Tovletoglou, L. Mukhanov, H. Vandierendonck, D. S. Nikolopoulos, P. Lawthers, P. Koutsovasilis, M. Maroudas, C. D. Antonopoulos, C. Kalogirou, N. Bellas, S. Lalis, S. Venugopal, A. Prat-Perez, A. Lampropulos, M. Kleanthous, A. Diavastos, Z. Hadjilambrou, P. Nikolaou, Y. Sazeides, P. Trancoso, G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, D. Gizopoulos, and S. Das, "An Energy-Efficient and Error-Resilient Server Ecosystem Exceeding Conservative Scaling Limits", ACM/IEEE Design, Automation, and Test in Europe (DATE 2018), Dresden, Germany, March 2018.

[166] K. Tovletoglou, L. Mukhanov, G. Karakonstantis, A. Chatzidimitriou, G. Papadimitriou, M. Kaliorakis, D. Gizopoulos, Z. Hadjilambrou, Y. Sazeides, A.

Lampropulos, S. Das, P. Vo, “Measuring and Exploiting Guardbands of Server-Grade ARMv8 CPU Cores and DRAMs”, IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2018), Luxembourg, June 2018.

[167] Amazon, “Amazon Web Services, EC2 Reachability Test”, http://ec2-reachability.amazonaws.com

[168] Ballas, Camille, et al. "Performance of video processing at the edge for crowd-monitoring applications." (2018).

[169] Ernst, Dan, Nam Sung Kim, Shidhartha Das, Sanjay Pant, Rajeev Rao, Toan Pham, Conrad Ziesler et al. "Razor: A low-power pipeline based on circuit-level timing speculation." In Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture, p. 7. IEEE Computer Society, 2003.

[170] Bacha, Anys, and Radu Teodorescu. "Dynamic reduction of voltage margins by leveraging on-chip ECC in Itanium II processors." In ACM SIGARCH Computer Architecture News, vol. 41, no. 3, pp. 297-307. ACM, 2013.

[171] Papadimitriou, George, Athanasios Chatzidimitriou, Manolis Kaliorakis, Yannos Vastakis, and Dimitris Gizopoulos. "Micro-viruses for fast system-level voltage margins characterization in multicore CPUs." In 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 54-63. IEEE, 2018.

[172] Papadimitriou, George, Manolis Kaliorakis, Athanasios Chatzidimitriou, Charalampos Magdalinos, and Dimitris Gizopoulos. "Voltage margins identification on commercial x86-64 multicore microprocessors." In 2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 51-56. IEEE, 2017.

[173] Gupta, Meeta Sharma, et al. "Towards a software approach to mitigate voltage emergencies." Proceedings of the 2007 international symposium on Low power electronics and design. ACM, 2007.

[174] Gupta, Meeta S., et al. "An event-guided approach to reducing voltage noise in processors." 2009 Design, Automation & Test in Europe Conference & Exhibition. IEEE, 2009.

[175] Reddi, Vijay Janapa, and Meeta Sharma Gupta. "Resilient architecture design for voltage variation." Synthesis Lectures on Computer Architecture 8, no. 2, 2013: 1-138.

[176] Pan, Songjun, Yu Hu, Xing Hu, and Xiaowei Li. "A cost-effective substantial-impact-filter based method to tolerate voltage emergencies." In 2011 Design, Automation & Test in Europe, pp. 1-6. IEEE, 2011.

[177] John L. Henning, “Spec cpu2006 benchmark descriptions,” SIGARCH Comput. Archit. News, vol. 34, pp. 1–17, Sept. 2006.

[178] Singh G., Favor G. and Yeung A., "AppliedMicro X-Gene2," 2014 IEEE Hot Chips 26 Symposium (HCS), Cupertino, CA, 2014, pp. 1-24.

[179] SPEC CPU2017 home page: www.spec.org/cpu2017

[180] C. Bienia, S. Kumar, J. P. Singh, and K. Li, “The parsec benchmark suite: Characterization and architectural implications,” in Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, PACT ’08, (New York, NY, USA), pp. 72–81, ACM, 2008.

[181] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, D. Gizopoulos, G. Favor, K. Sankaran and S. Das, “A SystemLevel Voltage/Frequency Scaling Characterization Framework for Multicore CPUs”, IEEE Silicon Errors in Logic – System Effects (SELSE 2017), Boston, MA, USA, March 2017.

[182] Changyong, F. E. N. G., W. A. N. G. Hongyue, L. U. Naiji, C. H. E. N. Tian, H. E. Hua, and L. U. Ying. "Log-transformation and its implications for data analysis." Shanghai archives of psychiatry 26, no. 2 (2014): 105.

[183] Yang, Ruipeng, Dan Qu, Yekui Qian, Yusheng Dai, and Shaowei Zhu. "An online log template extraction method based on hierarchical clustering." EURASIP Journal on Wireless Communications and Networking 2019, no. 1 (2019): 135.

[184] Papadimitriou George, Athanasios Chatzidimitriou, and Dimitris Gizopoulos. "Adaptive Voltage/Frequency Scaling and Core Allocation for Balanced Energy and Performance on Multicore CPUs." In 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 133-146. IEEE, 2019.

[185] Hadjilambrou, Zacharias, Shidhartha Das, Paul N. Whatmough, David Bull, and Yiannakis Sazeides. "GeST: An Automatic Framework for Generating CPU

Stress-Tests." In 2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 1-10. IEEE, 2019.