Edge and Cloud Provider Cost Minimization by Exploiting Extended Voltage and Frequency Margins

Christos Kalogirou[A], Panos Koutsovasilis[A], Manolis Maroudas[A], Christos D. Antonopoulos[A], Spyros Lalis[A], and Nikolaos Bellas[A]

[A]Department of Electrical and Computer Engineering, University of Thessaly

{hrkalogi, pkoutsovasilis, emmmarou, cda, lalis, nbellas}@uth.gr

Abstract

Energy costs contribute significantly to the total cost of operation for cloud and edge infrastructure providers. Both conventional (Voltage and Frequency Scaling) and more aggressive (undervolting, overclocking) techniques can be applied to reduce the energy footprint of the infrastructure and thus the cost of the operator. However, these techniques may affect the quality of service (QoS), potentially activating service level agreement (SLA) violation penalties for the provider. In this paper we model and study this tradeoff. We find that undervolting is the most effective of the three techniques in reducing infrastructure operation cost. We identify optimal operating points, and we study the effect of different parameters, such as the severity of SLA penalties, the length of the job and the existence of error protection mechanisms, on the optimal operating point and the extent of potential benefits.

1 Introduction

Minimizing energy consumption has become a critical concern in computing, due to technical, economic and environmental reasons. For cloud/datacenter deployments, energy corresponds to a significant percentage of the operation cost and the environmental footprint of the facility. At the same time, power dissipation is the main factor limiting the denser packing of servers. In edge deployments, technical constraints often introduce even tighter power and energy limitations.

Hardware manufacturers have introduced Voltage and Frequency Scaling (VFS) [10] and more recently power throttling as effective mechanisms of power management. VFS enables the concurrent manipulation of supply voltage (V_{dd}) and operating frequency (f) among different nominal (V_{dd}, f) tuples. VFS can result to significant power savings, which however come at the cost of lowering the frequency (and thus typically performance) of the CPU. Due to the associated performance loss in computational workloads, VFS does not necessarily reduce the energy footprint of the workload.

Another, more aggressive, power reduction technique is voltage overscaling (undervolting) [2]. In this case, the CPU is supplied with lower voltage than the nominal for the respective frequency. This technique relies on the fact that manufacturers have introduced extensive guardbands to their chips to guarantee stable operation under adverse – and very improbable – combinations of conditions. A complementary approach is frequency overscaling (overclocking). In this case the chip is clocked at a frequency higher than the nominal for the respective supply voltage. Overclocking increases power consumption, however at the same time it increases performance. Therefore, it still presents opportunities for energy savings.

All three techniques may have the potential to affect the stability of the system, whereas two of them (VFS and overclocking) also affect the performance. Voltage and frequency overscaling make the processor more vulnerable to timing errors. VFS can also affect system reliability, although in a less profound way; operating to lower voltage/frequency steps makes CPU cells more vulnerable to lower energy transient upsets, like the effects of cosmic particles. As a result, the quest to reduce energy consumption may affect the quality of service (QoS) provided by the system. Cloud and edge operators are contractually bound to a certain QoS to their users, through Service Level Agreements (SLAs) [6]. Violation of the SLA is typically associated to penalties for the operator, reducing its profit margins.

In this paper we follow a modeling approach to evaluate the feasibility and the associated tradeoffs, and try to identify configurations for maximizing the profit margins of Cloud and Edge resources operators by applying the aforementioned techniques. For all three approaches we estimate the cost for the operator due to energy consumption and potential SLA violation penalties and study its sensitivity to different parameters. To the best of our knowledge this is the first effort to study the operation of edge / cloud nodes outside nominal configuration from a cost minimization perspective.

The rest of the paper is organized as follows: Section 2 introduces the cost estimation model for each of the three aforementioned techniques. Section 3 applies the models on an assumed edge/cloud setup and evaluates opportunities and tradeoffs. Section 4 outlines related work. Finally, Section 5 concludes the article.

2 Cost Modeling and Minimization

This section introduces the models that estimate the operational cost for the cloud/edge provider due to the energy and SLA violation penalties. In order to reduce energy we use both conventional VFS and the more aggressive reduction of voltage and frequency safety margins introduced by hardware manufacturers.

2.1 Assumptions

We try to minimize energy costs by reducing the voltage beyond the nominal point for the respective frequency (undervolting), increasing the frequency (and thus the per-

Table 1: Summary of variables and parameters

Variable	Description
d	Constant. Failure sensitivity to frequency scaling (within nominal range)
$delay$	Task execution delay over contractual (SLA) agreement
C	Constant. Average switching capacitance of the processor
E_{max}	Energy consumption at highest operating point
E_e	Expected energy consumption when considering potential failures
E_x	Energy consumption at a (V_x, f_x) operating point
f_{max}	Constant. Processor frequency at the highest performing nominal operating point
f_{min}	Constant. Processor frequency at the lowest performing nominal operating point
f_x	Processor operating frequency
f_{PoFF}	Constant. Frequency at the Point of First Failure
$MTTF$	Mean time to failure
num_{ins}	Number of instructions of the task
P_{static}	Processor static power
p_{fail}	Failure probability
$price_{energy}$	Constant. Energy cost per Joule
$price_{vm}$	Constant. Price per VM per time unit
$rw_{fraction}$	Fraction of the task that was executed before a failure
t_{max}	Task execution time at the highest performing nominal operating point
t_e	Expected task execution time when considering potential failures
t_x	Task execution time at a (V_x, f_x) operating point
V_{max}	Constant. Processor supply voltage at the highest performing nominal operating point
V_{PoFF}	Constant. Processor supply voltage at the Point of First Failure
V_{th}	Constant. Processor threshold voltage
V_x	Processor supply voltage
α	Constant. Quantifies part sensitivity to undervolting/overclocking (linear part)
β	Constant. Controls the relation between nominal voltage and the respective frequency
γ	Constant. Quantifies part sensitivity to undervolting/overclocking (exponential part)
κ	Intensity of SLA violation penalty wrt. to user paid cost for the service
λ	Failure rate
λ_0	Constant. Failure rate at the highest performing nominal operating point

formance) of the processor beyond the nominal point for the respective voltage (overclocking), or by concurrently varying voltage and frequency, however at nominal (V_{dd}, f) points (VFS).

Energy cost is assumed to be constant (does not vary in time). We consider a cloud/edge deployment consisting of a homogeneous set of compute nodes both in terms of resources and inherent resilience.

Each task has an *a priori* known lifetime. For the sake of simplicity we consider errors to manifest as task failures/crashes (we do not consider SDCs), therefore error detection does not incur extra cost. In case of a failure, the whole task is re-executed at highest nominal V_{max}, f_{max}) point.

Table 1 summarizes the constants and variables we use in the models.

2.2 SLA Violation Penalty

Cloud/edge providers are contractually bound (SLA) to provide a certain QoS. If this QoS is not met, they need to compensate the customer according to the terms of the SLA. We assume that the main QoS metric is whether a task is executed within a deadline. We adopt a model [6] which linearly correlates SLA violation penalty with the extent of delay, after the deadline, of task execution.

More specifically, Eq. (1) quantifies the penalty where $delay$ is the additional time required for execution after the deadline and $price_{vm}$ is the cost paid by the customer for the execution of each VM per time unit. Finally, κ is a constant specified in the SLA which connects the compensation penalty to the price paid by the customer for the service. Higher values of κ are beneficial for the provider and vice versa.

$$penalty = \frac{delay * price_{vm}}{\kappa} \tag{1}$$

In order to calculate the delay, we need both the terms of the SLA and the expected execution time (t_e) of each task when executed under the control of each of three energy reduction mechanisms. We assume that agreed deadline specified in the SLA is based on the execution of the task at the highest performing nominal operating point (V_{max}, f_{max}) plus some slack.

2.3 Error Modeling

The failure probability due to voltage/frequency (over)scaling follows an exponential distribution [4]. Eq. (2) quantifies this probability. t_x is the lifetime of the task and $MTTF$ is the mean time to failure.

$$p_{fail} = 1 - e^{-t_x/MTTF} \tag{2}$$

$MTTF$ can be calculated as the inverse of the failure rate λ: $MTTF$ ($MTTF = \frac{1}{\lambda}$). In the next sections we will discuss in further detail how λ is estimated for each of the three energy reduction methodologies we study.

When $MTTF$ is relatively large compared to t_x, one could statistically assume that failures happen on average in the middle of task lifetime. However, this is not necessarily true as the order of $MTTF$ approaches t_x, which is often the case as we move to less safe operating configurations. In this case, the fraction of the task ($rw_{fraction}$) that had been executed before the failure occurred and will thus have to be re-executed is given by Eq. (3) [3].

$$rw_{fraction} = \frac{MTTF}{t_x} + \frac{1}{1 - e^{t_x/MTTF}} \tag{3}$$

2.4 Energy Consumption Modeling

V_x and f_x are possible operating voltage and frequency settings,respectively, of the processor, not necessarily corresponding to a nominal operating point. The supply voltage and the respective frequency at the highest performing nominal operating point are V_{max} and f_{max}. When a task is re-executed due to a failure, we greedily opt to do so at the (V_{max}, f_{max}) operating point in order run the task at the maximum system performance and thus minimize the probability of an SLA violation.

Energy consumption E_x on a (V_x, f_x) configuration is given by Eq. (4),

$$E_x = (CV_x^2 f_x + P_{static})t_x \tag{4}$$

where C is the average switching capacitance of the processor. The first term ($CV_x^2 f_x$) in the parenthesis corresponds to the dynamic power of the processor, P_{static} is its static power and t_x is the execution time of the task when the processor is clocked at a frequency equal to f_x, for the case where no failures occur. Assuming CPI does not significantly change with clock frequency, which can be expected to be the case unless the task is memory intensive with bad cache locality, t_x can be calculated by Eq. (5):

$$t_x = \frac{num_{ins} * CPI}{f_x} \tag{5}$$

where num_{ins} is the total number of instructions executed by the task.

The expected energy consumption E_e for the execution of a task at an operating point (V_x, t_x) is calculated by Eq. (6):

$$E_e = (1 - p_{fail})E_x + p_{fail}(rw_{fraction}E_x + E_{max}), rw_{fraction} < 1 \tag{6}$$

where p_{fail} is the probability of failure of the task when executed at the specific configuration. The first term of the equation ($(1 - p_{fail})E_x$) corresponds to the contribution to the average expected energy E_e of executions where a failure did not occur. The second term quantifies the energy contribution of task executions which resulted to failure after a percentage equal to $rw_{fraction}$ of the task was executed. In this case, the task needs to be re-executed at the nominal (V_{max}, f_{max}) configuration, consuming E_{max} additional energy.

2.5 Energy Reduction Methodologies

This section discusses the three models that estimate the energy consumption of the tasks. We estimate the failure rate (λ) and expected execution time (t_e) for each model. Finally, we estimate the cost of the energy according to Eq. (7), where $price_{energy}$ is the energy cost (per Joule).

$$energy_{cost} = E_e * price_{energy} \tag{7}$$

2.5.1 Expected Energy Consumption - Voltage and Frequency Scaling (VFS model)

In VFS we concurrently manipulate frequency and voltage, however the processor always operates at nominal (V_x, f_x) points. We calculate the frequency f_x corresponding to each voltage level V_x using Eq. (8) [16]. β is a CPU model-specific constant and V_{th} is the threshold voltage.

$$f_x = \beta \frac{(V_x - V_{th})^2}{V_x} \tag{8}$$

We also estimate the failure rate according to Eq. (9), where λ_0 is the failure rate at maximum nominal performance operating point, d reflects the hardware proneness to failure when scaling voltage and frequency, and f_{max} and f_{min} are the highest and lowest nominal frequencies respectively.

$$\lambda = \lambda_0 e^{\frac{d(fmax - \beta(V_x - 2V_{th} + \frac{Vth^2}{V_x}))}{fmax - f_{min}}} \tag{9}$$

In VFS, error rates are correlated to the operating voltage and frequency, as in lower power/performance operating points circuits are more prone to transient faults due to lower-energy upsets (such as cosmic particles).

Finally, we have to calculate the expected execution time t_e according to Eq. (10). Given that we change frequency, the execution time is directly affected. We use t_x to represent the execution time at operating point (V_x, f_x). If a failure did not occur, the first term (t_x) provides the execution time of the task. However, if a failure occurred the first two terms estimate the execution time before the failure, while the third estimates the additional time that is required for re-execution (at the highest nominal operating point).

$$t_e = t_x - p_{fail}(1 - rw_{fraction})t_x + p_{fail}t_{max}, rw_{fraction} < 1 \tag{10}$$

2.5.2 Expected Energy Consumption - Voltage Scaling (VS model)

In this case, we reduce voltage while clocking the processor at the highest nominal frequency. We estimate the failure rate using Eq. (11) [14], where V_{PoFF} is the voltage corresponding to the Point of First Failure (PoFF) [5] and α and γ are constants we obtain by the data provided in [1] and [5]. The PoFF corresponds to the point at which the error rate equals 1 error every 10 million cycles. Beyond the PoFF error rate increases exponentially with the decrease of voltage [5], [1]. Timing errors are the dominant cause of failures when undervolting.

$$\lambda = \alpha 10^{\gamma(V_x - V_{PoFF})} \tag{11}$$

The execution time of the task is t_{max} for any V_x, as we do not change the frequency of the processor. Eq. (12) estimates the expected execution time and consists of two terms. The first term corresponds to the execution time at highest nominal operating point while the second one is the fraction of the task that was executed before the failure, should a failure occur. Only the first term contributes to execution time if a failure did not occur.

$$t_e = t_{max} + p_{fail}rw_{fraction}t_{max}, rw_{fraction} < 1 \tag{12}$$

2.5.3 Expected Energy Consumption - Frequency Scaling (FS model)

We now increase the frequency of the processor beyond f_{max}, while keeping voltage at the maximum nominal level V_{max}. The model is similar with the VS model. The mechanism of failures is also similar. However, in this case we estimate the failure rate using Eq. (13) in terms of frequency, instead of voltage, and f_{PoFF} is the frequency corresponding to the Point of First Failure.

$$\lambda = \alpha 10^{\gamma(f_{PoFF} - f_x)} \tag{13}$$

The expected execution time of the task can be calculated using Eq. (10).

2.6 Total cost reduction

We calculate the cost for the provider as the cost of energy plus the potential SLA violation penalty, as summarized by Eq. (14).

$$cost = energy_{cost} + penalty \tag{14}$$

We estimate the cost at any possible (V_x, f_x) point for each of the three mechanisms and compare it with the cost when executing at the highest performing nominal operating point (V_{max}, f_{max}).

3 Evaluation

For the evaluation of our models, we assume a processor clocked at 3.7 GHz (f_{max}) with highest nominal voltage 1.06 V. We also used the amazon pricing list[1] to select a realistic price for the VMs. More specifically, we chose Ireland as the region and the c4.xlarge VM type, which is compute optimized. The resulting VM cost for the client is 0.226$/hour. The energy cost for the provider is 0.07$/KWh [9].

We assume one task per VM and experiment with tasks with different execution times (1 second, 10 seconds, 100 seconds) and various values for κ (1, 2, 4). Moreover, we quantify the effects of protection mechanisms [14] which manage to identify and mask a percentage of sporadic faults.

For each model, we illustrate three figures. The leftmost ones represent the operator cost reduction which comes from the reduced energy consumption only, for tasks with different execution times. The figures in the middle quantify operator cost reduction due to reduced energy consumption, however also considering the potential penalties that need to be paid to the user due to SLA violations. We use tasks with an execution time of 100 seconds and we experiment with different values of κ. Finally, the leftmost figures illustrate the effect of fault protection/masking mechanisms to operator cost reduction. Again we assume tasks executing for 100 seconds. We also fix κ to 4. In all cases, the baseline operator cost is when executing the highest performing nominal operating point (V_{max}, f_{max}). Higher values in the vertical axis are better (correspond to higher cost reduction).

Comparing the three energy models, undervolting (VS) results to the highest cost reduction (13.31 % on average for the different scenarios depicted in Figure 1). VFS, as expected, is not very efficient with computational intensive tasks. It can only exploit the deadline slack over t_{max} to identify a nominal point with slightly lower energy footprint. However benefits are marginal as very soon the slack is consumed and SLA violation

[1]https://aws.amazon.com/ec2/pricing/

penalties outweigh any energy cost reduction. Overclocking (FS) has also limited effect on operator cost. It improves performance, thus reduces execution time, however at the same time it also increases power consumption. Although it is not effective in reducing cost, overclocking can improve operator profit as jobs leave the system faster, making it available to execute additional jobs and thus generate additional income for the operator. This is not evident in our diagrams, as they focus solely on the reduction of costs.

From Figures 1 and 2 it is obvious that as we approach the point of first failure ($V_{PoFF} = 0.901V$ and $f_{PoFF} = 4.2GHz$ respectively) massive errors appear and therefore the system operates at a non cost-effective configuration. The energy consumption increases due to failed tasks which are re-executed at the (V_{max}, f_{max}), after having earlier executed for some time in the undervolted (or overclocked) configuration. As we move to more aggressive voltage and frequency settings respectively, the cost stabilizes to the cost of a (V_{max}, f_{max}) execution, as the error rate is too high, tasks fail almost immediately and the total cost of execution essentially equals that of the re-execution at the highest performing nominal operating point (should the system overall survive the faults). It is clear that the goal of aggressive energy reduction mechanisms, such as VS and FS, should be to approach the PoFF, without risking to reach it. As a matter of fact, as the reader can observe in the leftmost figures, the optimal operating point corresponds, due to the cost of re-executions, to higher voltages (0.947 – 0.967 V) than V_{PoFF} and to lower frequencies (3.98 – 4.04 GHz) than f_{PoFF} for undervolting and overclocking respectively.

The middle figures depict the effect of SLA violation penalties to cost reduction. We assume the deadline of the task includes a 10 % slack over its expected execution time at the highest performing nominal operating point (t_{max}). For this slack, we observe that the three curves practically overlap, thus cost reduction is not sensitive to the values of κ we experimented with. However, if we decrease the slack so that the deadline approaches t_{max}, SLA violations are more frequent. As expected, in this case, higher values of κ (for example 4 in our experiments) are beneficial for the provider, as they limit the penalties owed to the user. The effect of the activation of SLA penalties to the cost-optimal operating points is also negligible (middle column diagrams) for both VS and FS. Once again, the dominant problem as we approach the PoFF is the exponential increase of faults, rather than the cost of SLA violations. On the other hand, we observe different optimal operation points for the VFS model when we apply the SLA. VFS model also decreases the frequency along with the voltage, hence the penalties start to occur even a few points below the highest nominal operating points.

For both VS and FS, the optimal operating points and the respective possible cost reduction depend on the execution time of the task according, to the leftmost figures. A lower execution time allows for more aggressive margins, as it reduces both the probability of a fault within the life of the task and the cost of the respective re-execution.

Finally, we study cost reduction when we apply protection mechanisms which can detect and mask the effects of a percentage of errors. According to [14], such mechanisms may mask up to 70 % of faults. As we observe, there is a limited improvement in the optimal operating points for VS and FS, which however does not translate to a measurable reduction of operator cost. However, those mechanisms are valuable in a different way: they provide a margin where errors may be detectable, yet not catas-

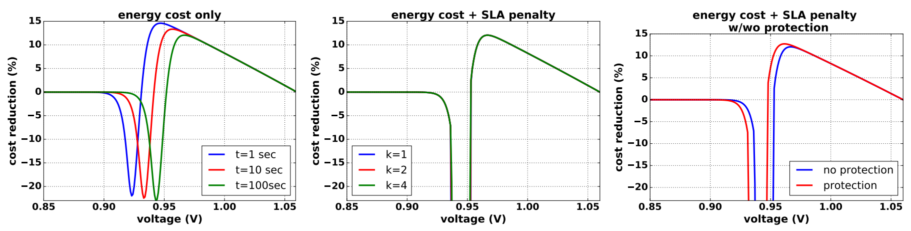

Figure 1: Operator cost reduction using Voltage Scaling

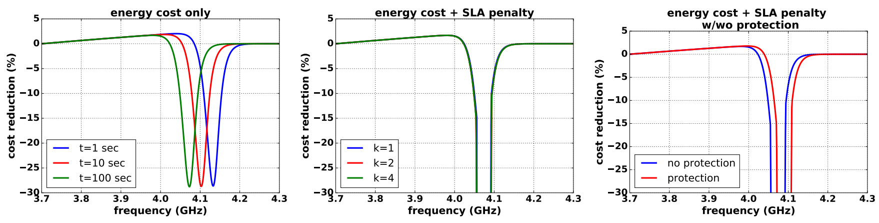

Figure 2: Operator cost reduction using Frequency Scaling

trophic for the system. In other words, they can provide observability to the fact that the system approaches the problematic range. For the VFS model, failure rate for nominal operating points is already very low, therefore protection mechanisms do not have any measurable impact.

4 Related work

Dynamic voltage and frequency scaling (DVFS) in cloud deployments has the potential to reduce power consumption and increase profit [11]. CloudScale [15] and GreenCloud [12] also apply DVFS to reduce the energy footprint of cloud applications and infrastructure. CloudScale can, in addition, predict potential errors. In our work we include voltage and frequency overscaling, two more aggressive techniques for saving energy. Although these techniques increase the probability of errors, we found that there is a range of operating configurations which can minimize the cost for the

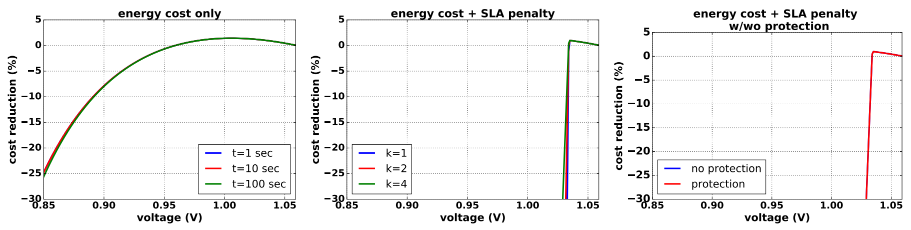

Figure 3: Operator cost reduction using Voltage and Frequency Scaling

cloud/edge provider, even when potential failures are taken into account.

The authors in [13] adopt dynamic voltage scaling to minimize energy consumption on high-performance computing systems. The model described in [2] proposes techniques that lower the supply voltage below nominal values, but also introduces mechanisms that allow the system to recover from the resulting timing errors and return to a stable state. Both these works try to maximize the energy gains but they do not consider the contractual QoS agreement (SLA) that the cloud/edge provider must ahere to. Eventually, due to timing errors, the profit of energy saving can be reduced by the SLA violation penalties.

The studies in [7, 8] introduce models that involve several QoS metrics, used in the SLAs between the consumers and providers through a complete environment for cloud applications. The resulting models focus on maximizing the provider profit while taking into consideration SLA violation penalties. These works miss the opportunity of further extending the infrastructure provider profit by exploiting either VFS or voltage overscaling. In our work, we adopt both voltage and frequency overscaling and we propose the respective models to estimate (and subsequently reduce) the cost to the Cloud and Edge operator. Moreover, we take into account the potential SLA violation penalties and highlight the optimal operating point (tradeoff between energy reduction and QoS) that increases operator profit.

5 Conclusions

In this paper we studied three models which quantify the cost reduction potential for cloud/edge operators by reducing the energy footprint of the infrastructure both with conservative (VFS) and more aggressive (VS, FS) techniques. All these techniques may affect the provided QoS. We take into account penalties due to potential violations of the contractually agreed QoS in cost estimation. Our study indicates that voltage overscaling is the most effective way to reduce the cost. Although, frequency overscaling does not reduce the cost significantly, it decreases the execution time of jobs and may allow for additional operator profit by executing more jobs within a given time window.

In this work we assume that nodes are homogeneous in terms of resources, energy consumption and inherent resilience. Moreover, we assume that the energy cost is constant in time and that we have only computational intensive tasks. However, real-world cloud and – especially – edge infrastructure and workloads are heterogeneous in all those aspects. Similarly, the cost of energy often varies in time. Taking into account all these parameters, paves the ground for interesting scheduling and resource management policies in cloud and edge deployments.

References

[1] D. Blaauw, S. Kalaiselvan, K. Lai, W.-H. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull. Razor ii: In situ error detection and correction for pvt and ser toler-

ance. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 400–622. IEEE, 2008.

[2] A. Cavelan, Y. Robert, H. Sun, and F. Vivien. Voltage overscaling algorithms for energy-efficient workflow computations with timing errors. In *Proceedings of the 5th Workshop on Fault Tolerance for HPC at eXtreme Scale*, pages 27–34. ACM, 2015.

[3] J. Daly. *A Model for Predicting the Optimum Checkpoint Interval for Restart Dumps*, pages 3–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[4] J. T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Gener. Comput. Syst.*, 22(3):303–312, Feb. 2006.

[5] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. A self-tuning dvs processor using delay-error detection and correction. *IEEE Journal of Solid-State Circuits*, 41(4):792–804, 2006.

[6] D. Dib, N. Parlavantzas, and C. Morin. Sla-based profit optimization in cloud bursting paas. In *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, pages 141–150. IEEE, 2014.

[7] D. Dib, N. Parlavantzas, and C. Morin. Sla-based profit optimization in cloud bursting paas. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 141–150, May 2014.

[8] S. Garg, S. Gopalaiyengar, and R. Buyya. Sla-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter. *Algorithms and Architectures for Parallel Processing*, pages 371–384, 2011.

[9] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM computer communication review*, 39(1):68–73, 2008.

[10] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, pages 38–43, Aug 2007.

[11] K. H. Kim, A. Beloglazov, and R. Buyya. Power-aware provisioning of cloud resources for real-time services. In *Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science*, page 1. ACM, 2009.

[12] D. Kliazovich, P. Bouvry, and S. U. Khan. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283, 2012.

[13] Y. C. Lee and A. Y. Zomaya. Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 92–99. IEEE, 2009.

[14] K. Parasyris, V. Vassiliadis, C. D. Antonopoulos, S. Lalis, and N. Bellas. Significance-aware program execution on unreliable hardware. *ACM Trans. Archit. Code Optim.*, 14(2):12:1–12:25, Apr. 2017.

[15] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 5. ACM, 2011.

[16] L. Tan, S. L. Song, P. Wu, Z. Chen, R. Ge, and D. J. Kerbyson. Investigating the interplay between energy efficiency and resilience in high performance computing. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 786–796. IEEE, 2015.