Exploiting CPU Voltage Margins to Increase the Profit of Cloud Infrastructure Providers

Christos Kalogirou, Panos Koutsovasilis,
Christos D. Antonopoulos, Nikolaos Bellas and Spyros Lalis
University of Thessaly
Volos, Greece
{hrkalogi,pkoutsovasilis,cda,nbellas,lalis}@uth.gr

Srikumar Venugopal and Christian Pinto
IBM Research
Dublin, Ireland
{srikumarv}@ie.ibm.com
{christian.pinto}@ibm.com

***Abstract*—Energy efficiency is a major concern for cloud computing, with CPUs accounting for a significant fraction of power consumption of datacenter nodes. CPU manufacturers introduce redundancy, partially in the form of supply voltage margins, to guarantee correct operation even for worst-case combinations of non-idealities in process variation and system operating conditions. However, these margins are unnecessarily wide for real-world execution scenarios, and merely translate to increased power consumption. In this paper, we investigate how such margins can be exploited by infrastructure operators in datacenters, by selectively placing client VMs on servers that are intentionally undervolted to save energy without compromising performance. This opportunity comes at the controlled risk of inducing failures and activating service-level agreement (SLA) violation penalties. We model the problem in a formal way, capturing the most important aspects that drive VM management and system configuration decisions. Then, we introduce XM-VFS, a VM scheduling and infrastructure configuration policy that reduces infrastructure operator costs by reducing voltage margins on datacenter nodes, and compare it with a state-of-the art policy which employs both dynamic voltage-frequency scaling (DVFS) and workload consolidation. We perform simulations for different scenarios and node configurations, and quantify the cost reduction for the infrastructure operator, taking into account the energy consumption and potential SLA violations. Our results show that significant gains, up to 17.05% and 16.1% energy and cost reduction respectively, can be achieved by configuring the nodes to operate at narrower voltage margins. In our simulations, we use realistic assumptions for voltage margins, energy consumption and performance degradation of applications due to frequency scaling, based on the characterization of commercial Intel- and ARM-based machines with diverse characteristics. Our model and scheduling policy are generic and scalable; they can be applied irrespective of the number and type of server nodes.**

***Index Terms*—extended margins, undervolting, energy efficiency, scheduling, cost effectiveness**

I. Introduction

Energy efficiency is a major concern for cloud computing, as energy corresponds to a significant fraction of infrastructure operators' costs. At the same time technical and environmental [1] reasons necessitate the reduction of the power consumption of datacenters.

On the software side, VM consolidation [2]–[4] is commonly used to reduce the power footprint of a datacenter. VM consolidation triggers VM migrations when necessary, aiming to allocate VMs on as few nodes as possible without performance loss, thus minimizing energy consumption.

On the hardware side, manufacturers have introduced Voltage and Frequency Scaling (VFS) [5] to reduce the power consumption of CPUs. VFS enables the joint manipulation of the CPU supply voltage (V_{dd}) and frequency (f) among different (V_{dd}, f) points. This way, a node can be configured to a more power-efficient operating point, by reducing both f and the respective V_{dd}, if the system identifies opportunities (slack) to do so without penalizing the performance of the workload below an unacceptable threshold.

At the same time, CPU manufacturers introduce voltage guardbands on top of the required voltage for each frequency step, in order to guarantee stable operation under extreme combinations of manufacturing variability, aging effects, workload stress, environmental conditions etc. These guardbands are conservative, as the worst case combination of those adverse conditions will occur very rarely, if at all, during the life of each chip. Hence there are significant opportunities for energy savings by reducing these guardbands in an educated manner [6], [7] and supplying the CPU with less voltage for the respective operating frequency (undervolting). However, even when applied judiciously, operation at extended margins may make the processor more vulnerable to errors, affecting – even if slightly – system stability and increasing the probability of failures, especially for scale-out deployments. Node instability may lead to violation of Service Level Agreements (SLAs), which, in turn, impose certain penalties and lead to reduced profits for the provider of the computing infrastructure.

In this work, we study the trade-off between the reduced energy cost due to operation at extended margins, and the SLA violation penalties due to the increased probability of node failures. The contributions of this paper are:

- We introduce an analytic and architecture-agnostic model, which captures the trade-off between cost minimization for computational infrastructure operators through energy footprint reduction – partially by reducing CPU voltage margins – and the risk of SLA violations (with the associated penalty costs).
- We introduce a novel scheduling and system configuration policy for cloud datacenters. Apart from VM

This work has received funding from the European Commission, project UniServer, contract number 688540.

consolidation and VFS, this policy exploits configuration at extended margins to minimize the energy footprint of cloud infrastructure. The policy takes into account the cost of potential SLA violation penalties due to failures, node overloading and VM migrations.
- We evaluate the proposed policy via simulation experiments, using realistic node profiles that are derived from real systems, and show that it achieves significantly better results compared to a state of the art policy.

For the purposes of our evaluation, we characterize two commercial server-class systems with different processor architectures, an Intel Xeon E3-1220 v5 and an ARM-based Ampere Computing X-Gene 3, to quantify their voltage margins, power consumption and computational capacity under different configurations. We take into account that a datacenter typically runs a variety of jobs with different performance characteristics (e.g., I/O-bound or CPU-bound applications) which do not behave similarly when applying VFS. To the best of our knowledge, this is the first effort to study the operation of a cloud infrastructure beyond the nominal configuration of server machines, using real hardware attributes.

The results from our experiments show that up to 17.05% higher energy gains at the plug can be achieved by configuring nodes at extended margins. This translates to up to 16.1% increased profits for the datacenter operator, despite the higher probability of node failures and SLA violations.

The rest of the paper is structured as follows. Section II captures the main aspects of the scheduling problem in a formal way. Section III introduces our scheduling and system configuration policy, which exploits the fact that servers can operate at extended margins in an educated manner, with a relatively low probability of failure. Section IV discusses how we have experimentally quantified the key parameters of server nodes used in our simulations. Section V focuses on the experimental evaluation of our approach and its comparison with a state of the art VM and node management policy. Section VI provides an overview of related work. Finally, Section VII concludes the paper.

II. Problem Modeling

This section introduces an analytic system model which captures the most important server configuration and VM scheduling aspects, when considering extended margins. We assume that nodes are homogeneous in terms of CPU hardware. It is straightforward to extend the model in order to capture CPU heterogeneity (not shown here for brevity).

A. Nodes and CPU operating points

We assume a datacenter with N nodes $n_i, 1 \leq i \leq N$. Each node n_i has finite memory Mem_i. The node's CPU can operate in various nominal (V, f) points, where V is the supply voltage and f is the frequency. Let the highest performance (and power) operating point be (V_{max}, f_{max}). When the CPU runs at a nominal operating point, it executes code with a negligible (zero) probability of a failure.

For each nominal operating point (V, f), the CPU can operate at a sub-nominal and more energy-efficient point (V_x, f) where $V_x < V$, exploiting the available voltage margin. Despite carefully selecting a safe (V_x, f) pair, we pessimistically assume that execution at extended margins can lead to a failure with probability $Pfail_{V_x,f}$. We determine the extended margins points and the respective failure probabilities by experimental characterization on real hardware (see Section IV).

B. Application workload – VMs

Each application is packaged as a separate virtual machine (VM). The notion of a VM serves here purely as an abstraction for isolated/contained execution. Let there be M different application VMs $VM_m, 1 \leq m \leq M$, which are submitted for execution on the server farm. Each VM has different resource requirements, defined in a service level agreement (SLA) to which the provider commits when accepting a VM.

Let Mem_m^{SLA} be the amount of memory requested by VM_m, specified in the SLA. We assume that the requested memory is allocated in full before a VM starts running, and remains allocated during the entire lifetime of the VM. The memory serves as a proxy for any static resource requirement.

Let $CPU_{m,f_{max}}^{SLA}$ be the upper bound for the CPU computational capacity that will be requested by VM_m, also defined in the SLA. This is expressed as the fraction of the full CPU computational capacity at the maximum frequency f_{max}. However, the VM typically does not need this capacity at all times. Let $CPU_{m,f_{max}}^{req} \leq CPU_{m,f_{max}}^{SLA}$ be the CPU capacity that is requested by VM_m at runtime, again expressed as a fraction of the full CPU capacity at the maximum frequency.

The performance of a VM depends on the frequency of the CPU. However, compute-intensive applications are more sensitive to frequency scaling than memory- or I/O-intensive applications. The same applies to the CPU capacity requirements of a VM: if the CPU frequency drops below f_{max}, the effective total CPU capacity is reduced, hence the relative CPU capacity (percentage of total) that is required by a VM to achieve the same performance effectively increases. The degree of this increase depends on the performance sensitivity of the VM to frequency scaling. To capture this effect, we transform the requested CPU capacity at f_{max} to the equivalent CPU capacity for any other frequency f, and use the shorthand notation $CPU_{m,f}^{req}$ to refer to this value. Note that $CPU_{m,f}^{req} \geq CPU_{m,f_{max}}^{req}$. We determine the performance sensitivity to frequency scaling for different application classes via experimental characterization on real hardware (see Section IV).

C. VM scheduling

Scheduling is done periodically. Each period has fixed length equal to K timeslots of $slotT$ time units each. At the end of each scheduling period, the scheduler determines (a) the mapping of newly arriving VMs, as well as potential migrations of existing VMs to nodes, and (b) the (V, f) setting of each node.

Let (V_i, f_i) be the selected (nominal or extended) operating point for node n_i. Also, let the assignment of VMs to nodes for the next period be encoded via the hosting matrix H, where $H[i,m] = 1$ if node n_i hosts VM_m, else $H[i,m] = 0$. This matrix represents the mapping of VMs to nodes for the respective period. In order for a placement to be valid, a node n_i must have enough memory to accommodate the VMs assigned to it: $Mem_i \geq \sum_{m=1}^{M} H[i,m] \times Mem_m^{SLA}$.

Let Mig_m be the number of timeslots needed to migrate VM_m to node n_i. We assume that the VM remains unavailable during migration. Without loss of generality, we assume that VM migrations take at least one timeslot, do not create any cost on the receiving node, and finish fast, i.e., $1 \leq Mig_m \ll K$. We use matrix X to model the migration overhead. The last dimension of X corresponds to timeslots within the current period. $X[i,m,k] = 1$ if VM_m runs on n_i in the kth timeslot, otherwise $X[i,m,k] = 0$. Therefore, for a migration overhead of Mig_m timeslots, $X[i,m,k]$ will be 0 for $0 \leq k \leq Mig_m - 1$ and $X[i,m,k]$ will be 1 for $Mig_m \leq k < K$.

D. CPU load and CPU capacity allocation to VMs

The CPU load of node n_i in timeslot k is equal to the total effective requested CPU capacity of all VMs that are running on the node: $Load_i^k = \sum_{m=1}^{M} X[i,m,k] \times CPU_{m,f_i}^{req}$. Note that while we assume each VM to have steady CPU requirements during a period, the load may change within a period due to migrating VMs which start running on the node with some delay (after the migration completes).

The CPU capacity allocated to each VM during a given timeslot depends on whether the node is under- or over-loaded:

$$CPU_{i,m}^{alloc,k} = \begin{cases} X[i,m,k] \times CPU_{m,f_i}^{req} & \text{if } Load_i^k \leq 1 \\ X[i,m,k] \times \frac{CPU_{m,f_i}^{req}}{Load_i^k} & \text{if } Load_i^k > 1 \end{cases} \quad (1)$$

If a node is under-loaded during the given timeslot, every VM will get the requested CPU capacity, else the CPU capacity will be shared among VMs proportionally to their requests. Here we assume equal priorities, but it is straightforward to consider priority-based scheduling. The multiplication with $X[i,m,k]$ is so that $CPU_{i,m}^{alloc,k} = 0$ in case VM_m is under migration during the given timeslot.

E. Failures due to operation at extended margins

A node that operates at an extended operating point (V_i, f_i) will fail in a scheduling period with probability $Pfail_{V_i,f_i}$. To focus on the essence of the problem, we adopt a simple failure model. We assume that every failure leads to a node crash, affecting all VMs that are hosted on the node. Also, failures occur at the beginning of the period, and the failed node remains unavailable during the entire period. We assume that node failures are transient and non-systematic: a failed node recovers and is able to host VMs by the next period. The latter is a realistic assumption provided that the scheduling period is relatively large.

The VMs hosted at a failed node are not re-scheduled to another node during the period where the failure occurs, so they do not run at all during the entire period. Finally, we assume that application VMs are stateless, and can be restarted/re-scheduled at the next period on any node without any delay.

F. SLA violation cost

A VM may not always get the requested CPU capacity at every timeslot, either because it is under migration, because the host node is overloaded, or because the host node has suffered a failure. In all these cases, the provider pays an SLA violation penalty to the VM owner.

The SLA violation cost is modeled similarly to the approach presented in [8]. More specifically, we assume that it is equal to the product of the normalized extent of the violation (expressed as the difference between the provisioned CPU resources adjusted to the computational capacity of the CPU at frequency f_i CPU_{m,f_i}^{SLA} and the CPU resources that were actually allocated to the VM CPU_{m,f_i}^{alloc}, divided by CPU_{m,f_i}^{SLA}), and the duration of the violation. For any given timeslot k, this can be expressed as

$$SLAV_{i,m}^k = q_m \times \frac{CPU_{m,f_i}^{SLA} - CPU_{m,f_i}^{alloc,k}}{CPU_{m,f_i}^{SLA}} \times SlotT \times Price_{VM_m} \quad (2)$$

where q_m is a multiplier that increases the severity of the SLA penalty according to the importance (priority) of the VM VM_m, and $Price_{VM_m}$ is the price charged to the client for executing VM_m for a time unit as agreed in the SLA.

So, the SLA violation cost over an entire period for all VMs running on node n_i that does not fail, can be expressed as

$$SLAV_i^{nofail} = \sum_{k=1}^{K} \sum_{m=1}^{M} X[i,m,k] \times SLAV_{i,m}^k \quad (3)$$

Note that if n_i fails, none of the hosted VMs will get any CPU capacity. In this case, the total SLA violation cost for all VMs that are hosted on n_i over the entire period is

$$SLAV_i^{fail} = K \times SlotT \times \sum_{m=1}^{M} H[i,m] \times q_m \times Price_{VM_m} \quad (4)$$

G. Energy costs

Let $Power(V, f, u)$ be the power consumed by a node at operating point (V, f) when the CPU utilization is u. We determine this function via experimental characterization of real hardware (see Section IV).

The CPU utilization at timeslot k is given by

$$u_i^k = \min\{Load_i^k, 1\} \quad (5)$$

Assuming a fixed unit price $UnitPrice$ for energy, the total energy cost for node n_i over an entire period is

$$Energy_i = UnitPrice \times \sum_{k=1}^{K} Power(V_i, f_i, u_i^k) \times SlotT \quad (6)$$

Note that if n_i is not assigned any VMs, it will remain idle and its utilization u will be 0. Depending on whether idle nodes are shut-down or merely enter a low-power state, $Power(V, f, 0)$ may return zero or a non-zero value. Also, we assume that failed nodes do not consume any energy.

H. Overall operating cost estimation

Based on all the above, we can estimate the cost for operating node n_i for the next scheduling period, based on the scheduled placement of VMs encoded in H and X and assuming the node is configured to operate at (V_i, f_i), as

$$Cost_i = Pfail_{V_i,f_i} \times SLAV_i^{fail} + (1 - Pfail_{V_i,f_i}) \times (SLAV_i^{nofail} + Energy_i) \quad (7)$$

Either the node will fail and the provider will pay the SLA violation cost for the VMs that are hosted on the node for the full scheduling period, or the node will not fail and the provider will pay some SLA violation costs if the node is overloaded, plus the cost of the energy consumed by the node to run the VMs. This equation is an important tool when looking for the optimal operating point of each node for the next period. It considers the energy consumption, as well as the cost of any potential failures or overloads of the node.

The total estimated cost for the next period over all nodes is $\sum_{i=1}^{N} Cost_i$. Our objective is to devise policies that take this cost into account in order to reconfigure nodes and schedule the VMs on nodes in an educated manner, by exploiting the extended CPU margins to save energy as long as this outweighs the potential cost due to the increased probability of SLA violations as a result of failures, node overloads and VM migrations.

III. VM Scheduling and System Configuration Policy

We introduce *XM-VFS*, a policy based on the model discussed in Section II. The policy targets cost minimization for the cloud infrastructure provider. It quantifies the trade-off between energy consumption reduction and the cost of potential SLA violation penalties, applies a set of heuristics to select a mapping of VMs to nodes and finally, selects node configurations that minimize the total cost. An important novelty of the policy is that it exploits operation at extended margins on top of the traditional arsenal of VFS and VM consolidation.

The policy is executed at scheduling period boundaries. Scheduling consists of 4 steps: a) detection of overloaded nodes and selection of candidate VMs to be migrated away from these nodes, b) mapping of these migrated VMs and any newly arriving VMs to their new hosting nodes, c) powering-off underutilized nodes if their VMs can be migrated away, and d) re-configuration of node (V, f) operating point if necessary. The first 3 steps are similar in nature to previous work on VM consolidation [3]. Actually we use the same heuristics as [3] for step (c). In the following discussion we focus more on steps (a), (b) and (d).

To determine the VM placement and node configuration for period p, *XM-VFS* first identifies what would have been the most power efficient operating point $(V, f)_{i,p-1}^{efficient}$ of each node n_i for the previous period $p-1$, using the model described in Section II. This step takes into account both node and VM characteristics.

Then, the resource requirements of each VM for the next period are predicted, based on the historic information for the last 10 periods with exponential aging. Given this prediction we estimate the required computational capacity that corresponds to frequency $f_{i,p-1}^{efficient}$ for each node n_i. This step is described in lines 1-2 of Algorithm 1. Based on a static utilization threshold, $MaxUtil$, we then identify the nodes that are expected to become overloaded. If a node is overloaded, the least costly VMs are selected for migration away from the node, until node utilization is estimated to be less than $MaxUtil$ (lines 3-18 of Algorithm 1). Migration cost is a factor of migration overhead Mig_m, VM price $Price_m$ and VM priority q_m (lines 4-6). This step is performed at the beginning of each scheduling period, independently and concurrently on each node.

Algorithm 1 Node overload detection and VM selection for migration

Input: n_i, $vmList_{n_i}$
Output: VMsForScheduling

1: $f_{i,p} \leftarrow f_{i,p-1}^{efficient}$
2: estimate CPU utilization $u_{i,f_{i,p}}^{p}$
3: **if** $u_{i,f_{i,p}}^{p} > MaxUtil$ **then**
4: **for** VM_m in $vmList_{n_i}$ **do**
5: $Cost_m^{migration} \leftarrow Mig_m \times SlotT \times q_m \times Price_m$
6: **end for**
7: **while** $u_{i,f_{i,p}}^{p} > MaxUtil$ **do**
8: $Cost_{min} \leftarrow \infty$
9: **for** VM_m in $vmList_{n_i}$ **do**
10: **if** $Cost_m^{migration} < Cost_{min}$ **then**
11: $Cost_{min} \leftarrow Cost_m^{migration}$
12: $VMMinCost \leftarrow VM_m$
13: **end if**
14: **end for**
15: $VMsForScheduling.add(VMMinCost)$
16: re-evaluate $u_{i,f_{i,p}}^{p}$
17: **end while**
18: **end if**

Next, we map VMs that require (re)scheduling to nodes. The VMs that are selected for migration and newly arriving VMs during the same period are considered together for mapping to nodes. This is essentially a bin packing problem, with VMs corresponding to items and nodes corresponding to bins. We use the Best Fit Decreasing algorithm to allocate the VMs to the nodes. At this step, the actual allocation of the VMs takes place so that the load of each node n_i raises at most up to $MaxUtil$ assuming that each node n_i will operate at $(V, f)_{i,p-1}^{efficient}$. This step has linearithmic ($\mathcal{O}(m \log m)$) complexity w.r.t the total number of VMs to be (re)scheduled.

At this point, the algorithm uses the heuristic in [3] to identify opportunities to power-off nodes. For all nodes, starting from the least utilized one, it is evaluated whether all hosted VMs can be migrated to other nodes without overloading them. Potential destination nodes are identified using again the BFD heuristic. Assuming that the number of VMs per node is relatively small, the complexity of this step is $\mathcal{O}(n \log n)$, where n is the number of nodes in the datacenter.

The final step, described in Algorithm 2, is to find the optimal (V, f) configuration of each node, given the VM placement for the next scheduling period. The optimal node configuration minimizes overall cost at the node level, therefore for a fixed VM-to-nodes mapping the total cost for the entire datacenter as well, taking into account both energy consumption and potential SLA violation costs. We use Equation 7 in order to estimate the cost for each operating point (nominal and extended) and we choose the one that introduces the lowest cost. Algorithm 2 is executed independently on each node and its complexity is $\mathcal{O}(1)$ (equal to the number of operating points, nominal and extended, of the node).

Algorithm 2 Node configuration for cost minimization

Input: n_i, OperatingPointList
Output: Node n_i configuration $(V, f)_{i,p}^{efficient}$

$Cost_{min} \leftarrow \infty$
$(V, f)_{i,p}^{efficient} \leftarrow (V, f)_{i}^{max,nominal}$
for $(V, f)_{i,p}$ in $OperatingPointList$ **do**
  estimate $Cost_i$ at $(V, f)_{i,p}$
  if $Cost_{min} > Cost_i$ **then**
    $Cost_{min} \leftarrow Cost_i$
    $(V, f)_{i,p}^{efficient} \leftarrow (V, f)_{i,p}$
  end if
end for
configure node n_i at $(V, f)_{i,p}^{efficient}$

IV. Hardware Characterization

To base our evaluation on realistic values for the parameters quantifying the behavior of server nodes, we use real hardware platforms with commercially available CPUs used in modern servers. Table I summarizes the target systems used for this characterization. In the following, we explain how we determine the parameters that capture the behavior of a node in the model described in Section II and, specifically, (i) the extended margins operating points, (ii) the failure probability when operating at such points, (iii) the power estimation function for different operating points and utilization levels, and (iv) the impact of frequency scaling on the performance of applications.

A. Voltage margin characterization

In the Skylake processor, the voltage-frequency operating point is dynamically controlled by the P-State manager / DVFS governor. For our evaluation, we pick four frequency points at 3.3, 3.0, 2.5 and 2.0 GHz, which cover the range of the most common frequencies applied in the *balanced* mode of the governor. The average nominal supply voltages for these frequencies are 1147, 1075, 922, and 850 mV, respectively.

TABLE I: Target hardware platforms.

Parameter	Intel platform	ARM platform
Architecture	x86-64 (Skylake)	ARMv8 (Ampere X-Gene 3)
# of Cores	4	32
TDP (W)	80	125
Technology	14nm	16nm
Max Frequency	3.3 GHz	3.0 GHz
Cache Size	8 MB	32 MB
Memory	32 GB	128 GB

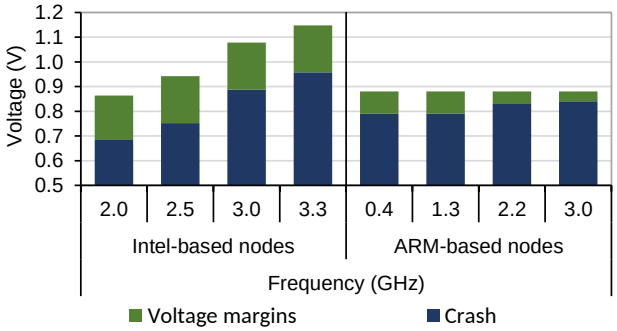

Fig. 1: Voltage margins characterization for the Skylake (*left*) and the X-Gene 3 CPUs (*right*).

The X-Gene 3 processor, targeting micro-servers, does not support DVFS. In nominal operation it allows only frequency scaling. We choose three frequency points, at 3.0, 2.2 1.3 and 0.4 GHz, covering the entire range that is supported by this CPU. In all cases, the nominal supply voltage equals 880 mV.

For each nominal operating point (V, f), we experimentally determine a corresponding extended margins operating point (V_x, f). We use 24 benchmark applications from the SPEC-CPU 2006 suite [9] that stress different CPU components. An experiment consists of running each application for a continuous period of 10 minutes (if needed, we run the application several consecutive times). We perform a series of experiments, where we keep the CPU frequency fixed to f, and gradually reduce the CPU voltage in steps of 10 mV, starting from the nominal voltage V. We stop when one of the experiments leads to any type of abnormal behavior (failure), and identify the immediately preceding voltage level (that did not result into a failure) as V_x for the frequency f.

Figure 1 illustrates the degree of undervolting that can be applied to each nominal operating point of the Skylake and X-Gene 3 CPUs. The top (green) area of the bars denotes the range of sub-nominal voltages that did not result into any failure. The lower (blue) area of the bars are the voltages that lead to failures for some or all applications. For the extended operating point at each frequency we choose the lowest voltage within the green area, i.e., the lowest sub-nominal voltage that did not lead to any failure. For the Skylake CPU, the sub-nominal voltages for the four frequency points of 3.3, 3.0, 2.5

and 2.0 GHz are 929, 865, 741 and 666 mV, respectively. For the X-Gene 3 CPU, the sub-nominal voltages for the three frequency points of 3.0, 2.2 1.3 and 0.4 GHz are 840, 830 790 and 790 mV, respectively. The characterization process took about 32 hours for both machines. While this amount of time is not negligible, such a characterization needs to be performed once, when adding a new node to the data center, and then only very sporadically to capture aging effects.

B. Failure probability

When modern CPUs operate at nominal voltage-frequency points, hardware failures are extremely rare [10]. Failures are expected to occur more frequently if the CPU operates at a non-nominal point.

In the above experiments when we operate the CPU at extended margins, we choose the largest sub-nominal voltage that does not lead to any failures. Afterwards, we validate the safety of the identified sub-nominal voltages by executing multiple experimental campaigns of random workloads at each extended operating point for 6 consecutive days each.

Being unreallistically pessimistic, we assume that the 6 days of execution time without any errors is the mean time to failure (*MTTF*), for all tested extended margin points. This corresponds to a failure rate of one failure every $518,400$ seconds, which – again pessimistically – we assume to be fatal, ignoring the possibility of correctable errors [7]. This failure rate is then used to estimate the probability of failure $Pfail_{V_x,f}$ within a scheduling period for a node that operates at any of the extended margin points (V_x, f). For example, assuming a scheduling period of 300 seconds, if a node is configured to operate at extended margins, the failure probability is $300/518,400 = 0.000579$. We use this $Pfail_{V_x,f}$ in our simulations to show that even with a relatively high failure probability, educated undervolting can provide a significant increase of the profit margin for the infrastructure provider.

C. Power consumption

We estimate the power consumption function $Power(V, f, u)$ for the two hardware platforms as follows: First, we run a series of experiments that exhibit different utilization levels, by varying the number of CPU cores used for numerous nominal voltage-frequency points. We record the power consumption, measured at the socket using a powerwall meter. Then, based on this data, we use linear regression to compute the parameters of the model $Power = A + B * u * V^2 * f$ used to estimate the power consumption as a function of CPU utilization u, supply voltage V and frequency f. This model serves as the power estimation function in our simulation experiments.

For the Skylake CPU, we get $A = 34.01$ and $B = 18.98$, which predicts power consumption with $R^2 = 0.98$ and a root mean square error (RMSE) of 4.40 for the unseen configurations at nominal operating points. For the X-Gene 3 CPU, $A = 52.48$ and $B = 34.38$, yielding $R^2 = 0.99$ and an RMSE of 2.73. The low RMSE indicates that the predictions of both models are close to the actual power consumption. We have also performed experiments for configurations at extended margins to verify the accuracy of the power estimation models.

D. Performance degradation due to frequency scaling

The performance of an application depends on CPU operating frequency, but some applications are more sensitive to frequency scaling. To quantify this in a realistic way, we measure the execution time of different applications for each of the frequency points considered in the two platforms. The applications we use in these measurements are 24 benchmarks from the SPECCPU 2006 [9], and the whole CloudSuite benchmark suite [11]. After extracting the performance profile for each application on each operating frequency point, we apply a K-Means clustering algorithm to form clusters of applications with common behavior.

Our experiments reveal three main clusters of applications with distinctly different behavior. As shown in Figure 2, these clusters exhibit similar performance degradation as a function of frequency scaling on both platforms. The first cluster, to which we refer as *compute-bound*, includes applications with performance degradation which scales almost linearly to CPU frequency reduction, with a ratio of approximately $1x$. The second cluster, referred to as *mixed*, is less affected by the frequency reduction, however aggressive frequency reduction leads to significant performance loss up to 76%. The third *memory-bound* cluster includes applications that are relatively insensitive to frequency reduction.

In our simulation experiments (described in Section V) we assume three equiprobable performance classes for the application VMs, corresponding to the three clusters identified above. Depending on the class of VM_m, we apply the corresponding performance scaling behavior to transform the requested CPU capacity $CPU^{req}_{m,f_{max}}$ at the maximum CPU frequency f_{max}, to the equivalent CPU capacity $CPU^{req}_{m,f}$ at a lower frequency f. It should be noted that in a real setup the frequency scaling sensitivity of a job can be easily estimated at execution time, by using hardware performance counters to quantify the CPU intensiveness of the job.

V. Experimental Evaluation

In this section we experimentally evaluate *XM-VFS*, considering the tradeoff between energy cost and potential SLA violation penalties due to crashes or migrations. We experiment with two node architectures, with diverse characteristics: an Intel-based platform, which, as discussed in Section IV, has wide voltage margins enabling significant energy gains, and an ARM-based platform which has narrow voltage margins.

We resort to simulations, as it is not feasible to obtain access to a large-scale installation, with administrator privileges; the latter are necessary in order to configure nodes at extended margins. However, our simulations are based on parameters experimentally quantified on real hardware and simulation results are, therefore, realistic. We also used real values for the energy cost [12] and the VM pricing [13]. The energy price for our experiments is $\$0.15$ per KWh and the VM pricing range,

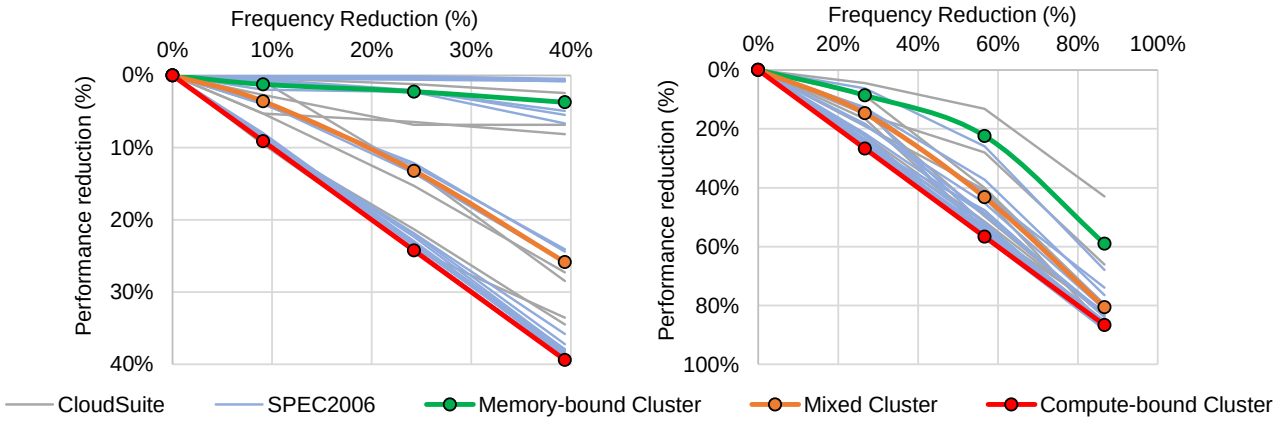

Fig. 2: Classes of applications, in terms of performance sensitivity to frequency reduction for Skylake (*left*) and X-Gene 3 (*right*).

for the types of VMs in our traces, is $\$0.0057 - \$0.4608$ per Hour.

Moreover, wherever applicable, we purposely made a set of pessimistic assumptions, therefore we expect gains to be higher on a real installation. More specifically: (i) We assume a baseline failure rate for nodes operating at extended margins, much higher than what we have observed during node characterization. In Section 4 we study the sensitivity of our policies on failure rate. (ii) We assume that all errors manifest as complete node crashes, incurring SLA violation penalties for all VMs on the node. (iii) We assume that all VM migrations result to VM downtime and thus also incur SLA violation penalties. Most virtualization and cloud management solutions support live migration, therefore the VM is available during migration.

We implemented our policies in the context of the CloudSim toolkit [14]. CloudSim is widely used, it already supports VM consolidation and it is configurable in terms of node hardware parameters. However, we had to extend it in the following ways:

- To support configuration at extended voltage margins and to simulate node crashes when operating at extended configurations.
- To be able to parse and use Google traces [15] as input. CloudSim previously could only use PlanetLab traces [16] which are rather small-scale.
- To implement the cost model discussed in II and our new policy.

For our experiments we used a subset of Google traces consisting of $10,000$ VMs. Each experiment simulates a time period of 1 day. We compare our policy against [17]. [17] is a recent work which optimizes the configuration of a datacenter using the VM consolidation techniques discussed in [3], in synergy with DVFS at the node level. The *LrMmt* (local regression minimum migration time) algorithm outperforms other policies for VM consolidation in [17], therefore we compare our policy against *LrMmt* in synergy with DVFS. We refer to this algorithm as *LrMmt-VFS* for the rest of the paper.

A. Target node utilization sensitivity analysis

One of the parameters controlling the behavior of resource and VM management policies in a datacenter is the target degree of node utilization. A low target utilization results to suboptimally utilized hardware and limits opportunities to power-off nodes (or alternatively to increase the throughput and efficiency of the datacenter). On the other hand, a high target utilization may result to node overloads, triggering SLA violations and necessitating VM migrations to mitigate overloads. Therefore, we performed a sensitivity analysis quantifying the performance of both our policy and *LrMmt-VFS* on nominally configured nodes, with different values for target node utilization.

Figure 3 illustrates the results of this analysis for our policy on both platforms. We present the total cost, the energy cost and the SLA violation cost for different node utilization targets. All values are normalized w.r.t. the highest total cost.

We observe that for both platforms a higher utilization target may indeed lead to SLA violations due to overloads and migrations. On the other hand, a lower utilization threshold keeps the nodes underutilized increasing, as expected, energy cost. It is clearly advisable to avoid extreme utilization targets. However, there is a range around the optimal utilization threshold, where the total cost is not significantly affected. Those observations, including the optimal values for each platform, are valid for the *LrMmt-VFS* policy as well. We conducted all the experiments using a utilization threshold target of 0.8 and 0.75 for the Intel and the ARM platform, respectively.

B. XM-VFS

Figure 4 illustrates the cost benefits of our policy (*XM-VFS*) w.r.t. *LrMmt-VFS* for Intel- (left) and ARM-based (right) nodes, respectively. We itemize costs as energy cost and SLA violation penalty cost due to VM migrations, node overloads

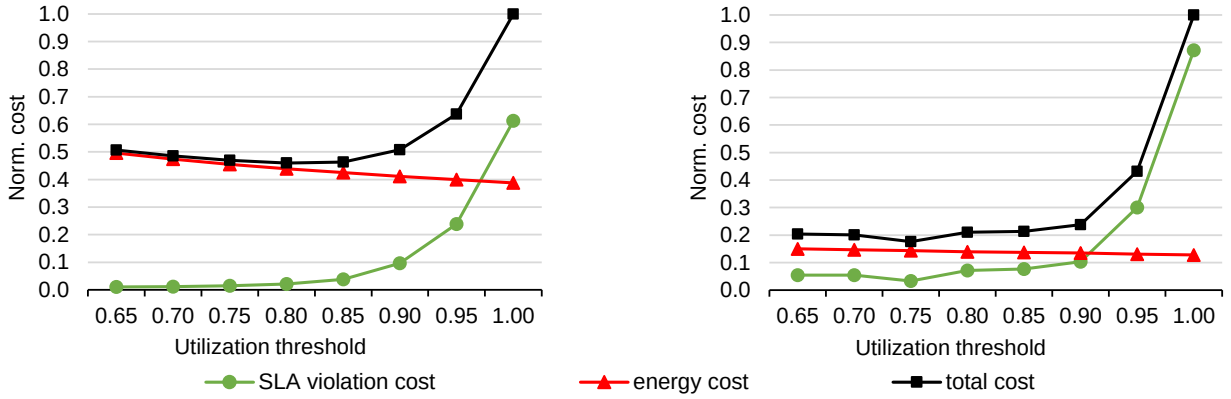

Fig. 3: Target utilization threshold sensitivity analysis for the Intel (left) and ARM (right) -based nodes.

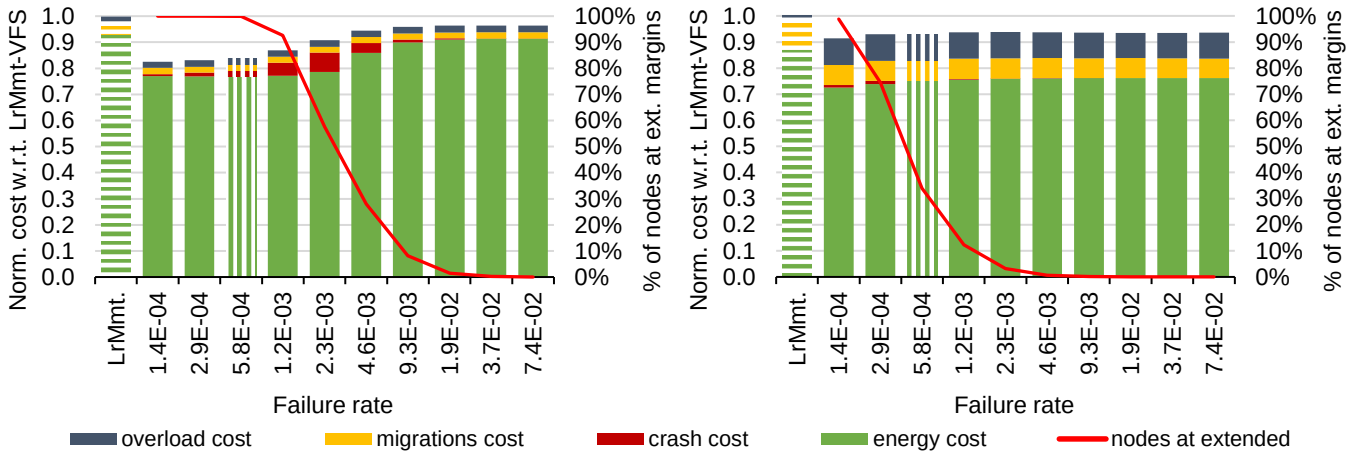

Fig. 4: Parametric analysis of cost w.r.t. failure rate. The horizontal axis is in logarithmic scale, with failure rate doubling between consecutive bars. Cost is normalized w.r.t. the cost achieved by LrMmt-VFS (leftmost bar, horizontally striped). The vertically striped bar (4th) corresponds to the baseline failure rate we assumed in the rest of the discussion in the paper. The red line indicates the percentage of the nodes that were configured at extended margins for each failure rate for the Intel (left) and ARM (right) -based nodes.

and node crashes. All costs are normalized w.r.t the total cost of the *LrMmt-VFS* policy for the corresponding platform. Figure 4 includes results with different failure rates. In this section, we will discuss the results of the baseline failure rate (vertical stripes). Crashes occur randomly in our simulated environment following the corresponding failure rate. In order to improve the statistical confidence of our results, we performed each experiment 10 times and we report averages among all runs.

The results indicate that *XM-VFS* successfully employs the cost function introduced in Section II to dynamically and adaptively react to the characteristics of the architecture and the workload and decide on datacenter configuration.

We observe that *XM-VFS* results to 16.1% lower total cost compared with *LrMmt-VFS* for Intel-based nodes. These cost savings come mainly from the exploitation of the extended margins of the Intel Xeon processor, which reduces the energy consumption of the nodes by 17.05%. We should note that we model and report energy consumption gains at the plug, rather than on the processor. This benefit is not outweighed by the cost of potential SLA violation penalties due to crashes which increases the total cost by 2.65%, when operating at extended margins. Therefore, *XM-VFS* is likely to configure nodes at extended margins – actually almost all the nodes are setup at a non-nominal configuration – despite the very pessimistic baseline failure rate we assume.

The overall cost reduction is less profound for the ARM-based platform. The X-Gene 3 processor has narrow voltage margins, therefore operation at extended margins does not offer energy savings high enough to significantly outweigh penalties due to SLA violations. As a result, our policy decides to configure at extended margins only 27.56% of the nodes. We observe a cost reduction 7.1% with *XM-VFS* compared with the *LrMmt-VFS* policy. Our policy manages to reduce costs by 6.38% by manipulating VMs better and selecting the most power efficient operating point of each node n_i more frequent than *LrMmt-VFS*, even on a nominally configured datacenter. If we allow node configuration at extended margins,

we observe another 0.72% cost reduction, taking into account both the energy savings and SLA violation costs due to node crashes. Note that, even on a datacenter with ideal X-Gene 3 nodes that would never fail when operated at extended margins and thus would all be undervolted, the expected energy benefit would be in the order of 5.5%, due to the narrow voltage margins of the ARMv8 in XGene-3.

At the pessimistic baseline failure rate we assume the percentage of node crashes throughout all experiments did not exceed 0.37% for any scheduling period, while during most periods there were no crashes at all. Also the number of the active nodes did not vary significantly across scheduling periods, resulting to approximately 516 and 452 active nodes for the Intel- and the ARM-based datacenter respectively, avoiding needless powering -on and -off of the nodes. This is enabled by the increased flexibility of computational capacity configuration offered by combining extended margins with built-in voltage-frequency (V, f) steps.

Another important observation is that – due to the aforementioned flexibility – *XM-VFS* manages to reduce the cumulative cost due to node overloads and VM migrations compared with *LrMmt-VFS*. There is obviously a tradeoff between migrations and node overloads. A policy may decide to tolerate overloads in order to reduce migrations, or opt for more migrations to reduce overloads. *XM-VFS* has more opportunities to reduce migrations, by operating at a higher performance (V, f) step whenever necessary, while at the same time limiting the cost increase associated with the higher (V, f) step by exploiting operation at extended margins. At the same time, the energy cost reduction creates a valuable cost margin for *XM-VFS*, giving it the opportunity to partially cover the cost of SLA penalties due to overloads in order to avoid even more costly VM migrations.

C. XM-VFS behavior at different failure rates

In the previous discussion we assumed a baseline pessimistic failure rate, when nodes operate at non-nominal configurations exploiting voltage margins. In this section, we evaluate the behavior of *XM-VFS* on both architectures, with different failure rates, both higher and lower than the baseline. The experimental results are illustrated in Figure 4.

The first important observation is that SLA violations due to crashes tend to decrease as we move to lower or higher failure rates. Low failure rates expectedly result to fewer node crashes. Higher failure rates, on the other hand, make *XM-VFS* configure less nodes (and beyond a point practically 0 nodes, as illustrated by the red line in Figure 4) at non-nominal settings, thus again decreasing crashes and the respective SLA violation penalties. On the ARM platform the improvement of energy efficiency by operating at extended margins is limited, therefore the policy generally configures less nodes in non-nominal settings at any given failure rate compared with the Intel-based datacenter.

The total cost also increases with the increase of the failure rate. The increase is gradual for Intel nodes. For ARM nodes it reaches its peak and stabilizes sooner (at a lower failure rate), as *XM-VFS* opts sooner to operate all nodes at nominal settings.

In any case, *XM-VFS* ubiquitously outperforms *LrMmt-VFS* for both platforms and at all failure rates.

VI. Related Work

All previous research efforts try to reduce datacenter energy consumption while operating the nodes at nominal conditions. However, undervolting [6], [7] the processor in a diligent way may offer more energy savings to the plug. As we discussed earlier, in a scale-out environment these energy savings come at the educated risk of increasing the failure probability, which may in turn lead to increased SLA violation penalties due to node crashes. No previous work studies the trade-off between energy savings and SLA violation penalties if the processor is configured at extended margins.

VM consolidation is a common technique used to improve energy efficiency. VMs are mapped to as few nodes as possible, in order to provide opportunities for unused nodes to be put to sleep state. However, consolidation may affect the quality of service (QoS) VMs enjoy to the extent of breaking contractual agreements (SLAs) and incurring penalties for the infrastructure operator. The problem of VM consolidation is further complicated by the dynamically changing workloads and resource requirements. The authors in [3], [4], [18] introduce heuristics to deal with this complexity when performing VM consolidation. Recent works [2], [19], [20] introduce optimized VM consolidation algorithms which also target the reduction of VM migrations. [21] is one of the few policies which, similarly to ours, takes into account that jobs which run in a datacenter have different characteristics (CPU- or IO-bound tasks).

Modern processors support DVFS, which enables operation of the processor at different nominal (V, f) operating points and different points of performance / power consumption. [22] introduces a VM allocation algorithm for a DVFS-enabled cluster. The authors in [23] exploit DVFS to reduce energy consumption for the execution of tasks with deadline constraints.

A number of recent research efforts combine VM consolidation with DVFS. This combination has the potential to provide significant energy gains, without significantly penalizing the QoS. Works in [17], [24] evaluate the benefits of applying VM consolidation in synergy with DVFS. They extend the policies in [3] in order to integrate DVFS. The authors in [17] report average energy savings up to 37.86% compared with previous work in [3]. As discussed in Section V, our policy outperforms [17].

To the best of our knowledge our work is the first that tries to minimize cloud infrastructure operator costs by carefully operating the processor outside its nominal operating envelope, taking the educated risk of node failures and the associated SLA violation penalties. On top of operation at extended margins, we also employ VM consolidation and DVFS. Moreover, we consider that VMs with different characteristics (CPU-

bound or IO-bound) behave differently when executed at different clock frequencies.

VII. Conclusions

Energy costs are a major component of operating costs of modern datacenters. This paper is the first to investigate whether undervolting of node CPUs can result to increased profit margins for cloud infrastructure providers. Undervolting, although educated, comes at the expense of possibly increased (even slightly) transient failure rates. We introduced a model which quantifies the tradeoff between energy cost reduction and the cost of SLA violation penalties due to transient node failures. We then discussed *XM-VFS*, an online VM scheduling and node configuration policy which uses the model to take decisions on node configuration (at nominal DVFS steps or at extended margins) and VM placement. We experimentally obtained real values for the parameters used in our model from an Intel Xeon-based server and the newest member (X-Gene 3) of the X-Gene family of Ampere Computing ARM-based high performance servers. The benefits of *XM-VFS* have been evaluated using a set of large-scale simulation campaigns. Our policy outperforms by 16.1% a state-of-the-art solution combining VM consolidation and DVFS.

Performance and resilience characteristics of different hardware parts can vary significantly, even for parts of the same family. This trend is expected to intensify as we move towards higher hardware densities and lower manufacturing geometries. Our work shows that instead of trying to battle against this heterogeneity through safe, yet inefficient margins, we should rather embrace it and treat it as an opportunity for increased efficiency.

References

[1] C. Ren, D. Wang, B. Urgaonkar, and A. Sivasubramaniam, "Carbon-aware energy capacity planning for datacenters," in *2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Aug 2012, pp. 391–400.

[2] Z. Cao and S. Dong, "An energy-aware heuristic framework for virtual machine consolidation in cloud computing," *The Journal of Supercomputing*, vol. 69, no. 1, pp. 429–451, 2014.

[3] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[4] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, no. 2, pp. 268–280, 2012.

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

[6] 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)*, 2017.

[7] A. Bacha and R. Teodorescu, "Using ecc feedback to guide voltage speculation in low-voltage processors," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2014, pp. 306–318.

[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*, pp. 371–384, 2011.

[9] J. L. Henning, "Spec cpu2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[10] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010, pp. 193–204.

[11] 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 *ACM SIGPLAN Notices*, vol. 47, no. 4. ACM, 2012, pp. 37–48.

[12] https://ec.europa.eu/eurostat/statistics-explained/index.php/Electricity_price_statistics.

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

[14] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[15] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," Google Inc., Mountain View, CA, USA, Technical Report, Nov. 2011, revised 2014-11-17 for version 2.1. Posted at https://github.com/google/cluster-data.

[16] https://github.com/beloglazov/planetlab-workload-traces.

[17] P. Arroba, J. M. Moya, J. L. Ayala, and R. Buyya, "Dynamic voltage and frequency scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 10, p. e4067, 2017.

[18] C. Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, May 2013, pp. 671–678.

[19] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Feb 2014, pp. 500–507.

[20] Y. Chang, C. Gu, F. Luo, G. Fan, and W. Fu, "Energy efficient resource selection and allocation strategy for virtual machine consolidation in cloud datacenters," *IEICE Transactions on Information and Systems*, vol. 101, no. 7, pp. 1816–1827, 2018.

[21] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A. A. Alelaiwi, and F. Li, "Minimizing sla violation and power consumption in cloud data centers using adaptive energy-aware algorithms," *Future Generation Computer Systems*, vol. 86, pp. 836–850, 2018.

[22] G. von Laszewski, L. Wang, A. J. Younge, and X. He, "Power-aware scheduling of virtual machines in dvfs-enabled clusters," in *2009 IEEE International Conference on Cluster Computing and Workshops*, Aug 2009, pp. 1–10.

[23] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, and X. Huang, "Enhanced energy-efficient scheduling for parallel applications in cloud," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*. IEEE Computer Society, 2012, pp. 781–786.

[24] E. Arianyan, H. Taheri, and V. Khoshdel, "Novel fuzzy multi objective dvfs-aware consolidation heuristics for energy and sla efficient resource management in cloud data centers," *Journal of Network and Computer Applications*, vol. 78, pp. 43–61, 2017.