2012 IEEE 4th International Conference on Cloud Computing Technology and Science

Lifetime or Energy: Consolidating Servers with Reliability Control in Virtualized Cloud Datacenters

Wei Deng, Fangming Liu, Hai Jin, Xiaofei Liao, Haikun Liu, Li Chen
Services Computing Technology and System Lab, Cluster and Grid Computing Lab
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China
{*wdeng, fmliu, hjin, xfliao*}*@hust.edu.cn*

***Abstract*—Server consolidation using virtualization technologies allow cloud-scale datacenters to improve resource utilization and energy efficiency. However, most existing consolidation strategies solely focused on balancing the tradeoff between service-level-agreements (SLAs) desired by cloud applications and energy costs consumed by hosting servers. With the presence of fluctuating workloads in datacenters, the lifetime and reliability of servers under dynamic power-aware consolidation could be adversely impacted by repeated on-off thermal cycles, wear-and-tear and temperature rise. In this paper, we propose a Reliability-Aware server Consolidation stratEgy, named *RACE*, to address *when and how to perform energy-efficient server consolidation in a reliability-friendly and profitable way*. The focus is on the characterization and analysis of this problem as a multi-objective optimization, by developing an utility model that unifies multiple constraints on performance SLAs, reliability factors, and energy costs in a holistic manner. An improved grouping genetic algorithm is proposed to search the global optimal solution, which takes advantage of a collection of reliability-aware resource buffering, and virtual machines-to-servers re-mapping heuristics for generating good initial solutions and improving the convergence rate. Extensive simulations are conducted to validate the effectiveness, scalability and overhead of *RACE* — in improving the overall utility of datacenters while avoiding unprofitable consolidation in the long term — compared with *pMapper* and *PADD* strategies for server consolidation.**

Keywords: Virtualization, cloud datacenter, energy management, reliability, consolidation, live migration.

I. Introduction

The fast proliferation of cloud computing services has promoted the massive datacenters. Cloud service providers consume many megawatts of power to operate such datacenters—such as Google ($> 1,120GWh$) and Microsoft ($> 600GWh$) [1]. However, a tremendous amount of such energy is actually over-provisioned for accommodating fluctuating workloads and peak demands. Servers in datacenters operate between merely 10 and 50 percent of their maximum utilization levels for most of their runtime [2].

Fortunately, dynamic workload consolidation among different servers based on virtualization technologies [3] has been extensively studied to enable datacenters to improve resource utilization and reduce power consumption. Specifically, all the *virtual machines* (VMs) hosting various applications are expected to be consolidated into a subset of *physical machines* (PMs) via VM migration [4], while other idle PMs (servers) can be switched to lower power states or shut down. However, profitable consolidation is not as trivial as conceptually packing the maximum number of VMs into the minimal number of PMs. There are a number of practical issues to be addressed, such as VM migration cost [3] [4] [5], resource contention and performance interference between co-located VMs [6] [7] [8], as well as cloud SLA violations [9].

While these existing studies have offered significant insights into the performance-energy tradeoff when performing consolidation in datacenters, the "flip side of the coin" with respect to *reliability loss* and *wear-and-tear of servers* due to over-aggressive consolidation [10] has not yet been well understood and alleviated. We argue that though greedy server consolidation policies can bring *short-term* energy savings, they may incur *long-term* costs on the reliability and lifetime of servers, due to repeated on-off cycles, wear-and-tear, and temperature rise.

First, due to workload fluctuations in datacenters, a certain set of servers will be selected by server consolidation to be shut down or switched to power-saving modes frequently. This can result in high transition frequency and on-off cycles that are recognized as the most crucial factors impairing disk reliability [11]. Furthermore, the on-off thermal cycle of processor contributes to another important factor causing fatigue failures [12] [13]. That is the rationale of why manufacturers usually limit the number of start/stop cycles of server disks to no more than $50,000$ over their entire lifetime, in order to guarantee the specified performance and reliability levels [2].

Second, after one or multiple rounds of server consolidation, a greater number of VMs running diverse application workloads will be packed into a less number of selected active servers. Both the utilization and temperature of such servers will increase, which may affect their lifetime. Reportedly, each $10\,^{\circ}\mathrm{C}$ temperature rise would reduce the server component life by 50% [14]. Temperature is an important influential factor in the reliability, performance and power consumption of modern processors and servers in datacenters [12] [15].

Therefore, frequent on-off cycles and high temperature can aggravate hardware failures, which are often recurrent according to recent empirical experiences: a crashed server due to faulty hardware is likely to crash again, incurring tremendous costs for procurement and replacement in large-scale datacenters [2]. These failures can incur partial or complete outage of services that cost about $\$5,000$ per minute [16]. Although Xie and Sun provided a reliability model for power-aware disk system [11], and recent studies considered the impacts of on-off cycles for power-aware server provisioning [2] [10] [17], few of them *jointly* minimize not only the impacts from on-off thermal cycles and temperature rise of servers, but also the costs from migration of VMs and state-switching

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

of PMs. Consequently, it could be risky and unprofitable to perform server consolidation for aggressive energy savings, without clearly understanding and answering the question: *when and how to perform energy-efficient server consolidation in a reliability-friendly and profitable way*?

In response, this paper presents a Reliability-Aware server Consolidation stratEgy, namely, *RACE*. It focuses on characterizing and analyzing the above problem as a multi-objective optimization, based on an utility model that we design to unify multiple constraints on performance SLAs, reliability factors, and energy costs — in a holistic manner. The model can estimate the profit of a new *configuration* of server consolidation (a mapping of VMs to PMs) with the greatest overall utility — by jointly minimizing the impacts from on-off thermal cycles and temperature rise, and the costs of VM migration and PM state-switching — while mitigating SLA violations due to the mismatch between demand for and supply of available server resources. Based on our model, an improved grouping genetic algorithm is proposed to search the global optimal solution, which takes advantage of a collection of reliability-aware resource buffering, and VMs-to-PMs remapping heuristics for generating good initial solutions and improving the convergence rate. We validate the effectiveness, scalability and overhead of *RACE* through extensive simulation experiments. Results show that RACE can improve the overall datacenters utility while avoiding unprofitable consolidation in the long term, compared with *pMapper* [5] and *PADD* [9].

II. Utility Models for Multi-objective Optimization of Server Consolidation

In this section, we construct our utility model for multi-objective optimization of server consolidation in virtualized datacenters, including the underlying assumptions. We model a datacenter with a set of VMs $V = \{v_1, v_2, \ldots, v_{|V|}\}$ running across a set of PMs (servers) $S = \{s_1, s_2, \ldots, s_{|S|}\}$. Without loss of generality, we consider a discrete-time model in which the time horizon ξ (*e.g.,* in a timescale of hours) is slotted into T equal time slots. The duration of one time slot is $\tau = \xi / T$. $\chi_{|V|\times|S|}(t)$ denotes the system *configuration* at time slot t. For example, for an element x_{ij} of χ, $x_{ij} = 1$ if PM s_j hosts VM v_i, otherwise $x_{ij} = 0$. In response to dynamic workloads, it is critical to decide *when* and *how* to perform consolidation is necessary and profitable at runtime. To quantify the potential benefits and costs of each new configuration, we develop utility models to capture essential aspects of datacenters: (1) performance SLA satisfaction $U_{SLA}(\chi)$ in term of server resources guarantee with respect to application demands, (2) reliability impact $U_r(\chi)$ due to on-off thermal cycles, wear-and-tear and temperature rise of PMs, (3) energy savings by consolidating VMs to a less number of PMs, and energy costs for migrating VMs and switching off PMs, jointly denoted as $U_e(\chi)$. Then, the overall utility $U(\chi)$ of a new configuration χ for the next time slot can be expressed as a weighed combination:

$$U(\chi) = \lambda_{SLA} U_{SLA}(\chi) + \lambda_r U_r(\chi) + \lambda_e U_e(\chi) \quad (1)$$

where λ_{SLA}, λ_r, and λ_e are the utility weights of performance SLA, reliability, and energy impacts, respectively, according to the design preferences of datacenter administrators.

Only when the overall utility $U(\chi)$ of a new configuration χ is positive, its corresponding server consolidation strategy is regarded as profitable and necessary. Accordingly, the objective of *RACE* is to guide consolidation by choosing the configuration with greatest overall utility, while avoiding unprofitable and aggressive reconfigurations. To this end, the first challenge that we shall address in the following subsections is: how can we construct detailed and practical models to quantify respective utilities of performance SLAs ($U_{SLA}(\chi)$), reliability impacts ($U_r(\chi)$), and energy costs ($U_e(\chi)$) in an unified economic viewpoint?

A. The Performance SLA Model

Suppose multiple VMs with fluctuating resource demands run on the same PM, the peak resource demands of these co-located VMs may exceed the capacity of PM, potentially leading to SLA violations. SLA violations is defined as the percentage of mismatch between demand for and supply of PM resources. Hence, we translate performance SLA as resource guarantees in multiple dimensions to the demands of VMs. Let P_o denote the probability of resource shortage, then the target is to ensure P_o not exceed a given SLA threshold p_{SLA}, expressed as Eq. (2):

$$P_o = Probability[\sum_{i=1}^{|V|} x_{ij} R_{ik}(t) \geq C_{jk}] \leq p_{SLA} \quad \forall v_i \in V, \forall s_j \in S, \forall t \in T, k = 0, 1, 2, 3 \quad (2)$$

where C_{jk} represents the capacity of resource k associated with PM s_j, $R_{ik}(t)$ denotes the demand for resource k of VM v_i at time slot t, and $k = 0, 1, 2, 3$ represent the multi-dimensional resources including CPU processor, memory, disk and network, respectively. The impact of consolidation on application performance is quantified by *SLA utility*: for a PM s_i, a monetary reward $R_{SLA}(i)$ for meeting the SLA target, whereas a monetary penalty $P_{SLA}(i)$ for missing it. With a new configuration χ, the SLA utility of datacenter is:

$$U_{SLA}(\chi) = \sum_{i=1}^{|S|} U_{SLA}(s_i) \quad (3)$$

$$where\ U_{SLA}(s_i) = \begin{cases} R_{SLA}(i), & \text{if } P_o \leq p_{SLA} \\ P_{SLA}(i), & \text{otherwise} \end{cases}$$

Since the resource requirements of VMs are time-varying, we need to estimate their demands at each time slot to allocate appropriate resources. We employ a commonly used ARMA (autoregressive moving average) [18] prediction technique. It predicts the workload demands $R^e_{ik}(t+1)$ of the next time slot based on the average of the w previously measured demands R^m_{ik} via the following equation:

$$R^e_{ik}(t+1) = (1-\lambda) \cdot R^m_{ik}(t) + \lambda \cdot \frac{1}{w} \cdot \sum_{l=1}^{w} R^m_{ik}(t-l) \quad (4)$$

where λ is used to weigh the current value against past historical measurements. As we will use in our experiments in Sec. IV, a typical setting is $w = 3$ and $\lambda = 0.5$ [10] to give balanced weights to the current and historical measurements.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

Then a question arises: *when to trigger the decision making of whether or not to perform consolidation?* We define a metric *Load* to characterize the multi-dimensional resource utilizations of a PM as the product of its CPU, memory, disk and network utilization ratios:

$$Load = \frac{1}{(1-\theta_{CPU}) \cdot (1-\theta_{Mem}) \cdot (1-\theta_{Disk}) \cdot (1-\theta_{Net})} \quad (5)$$

where θ_{CPU}, θ_{Mem}, θ_{Disk}, and θ_{Net} are the corresponding average utilization ratios ($0 \sim 100\%$) of the PM resources at each time slot. Such resource utilization will be set to $1-\varepsilon$ ($\varepsilon \to 0$) when the corresponding resource is fully utilized (100%), so as to avoid PMs with infinite *Load*. When the average *Load* of the datacenter is below a minimum threshold L_{min} (*i.e.,* $\sum_{i=1}^{|S|} Load_i/|S| < L_{min}$) for a duration of $T_{timeout}$, *RACE* will start to decide whether to perform consolidation or not, based on the calculated utility. To react quickly to workload variations while avoiding frequent reconfiguration overheads, one can set such a scheduling interval $T_{timeout}$ to the order of 10 minutes in practice [17].

B. The Reliability Gain and Cost Model

The *reliability utility* U_r depends on both reliability cost C_r and gain G_r. The reliability costs involve the wear-and-tear due to on-off thermal cycles of PMs to be turned off, and the impact from temperature rise of active PMs. We use the decrease of *mean time to failure* (MTTF) $\triangle MTTF$ to evaluate the reliability costs. In particular, as it is evidenced that disk and processor are the two main components that lead to server failures [12] [15], we focus on modeling the reliability costs of disk and processor in terms of wear-and-tear and temperature rise, respectively. On the other hand, turning off idle PMs can conserve their lifetime [19], which is quantified as the reliability gain G_r in next subsection. Then, the overall reliability utility of a configuration χ is calculated as $U_r(\chi, MTTF) = G_r(\chi) - C_r(\chi, \triangle MTTF)$.

1) ***Utility of reliability gain****:* We denote the *reliability utility per unit time* as π, which is computed as follows. First, we calculate the lifetime of a PM in term of MTTF, and the dollar cost per PM including procurement and repairs. We assume that PMs in our system have the same MTTF. Then, π is the ratio of the dollar cost to MTTF. For instance, suppose the average cost for a PM is $2,000$ dollars and its average lifetime is 5 years, then π is about 4.56 cents per hour. Let $y(t)$ be the number of active PMs at time slot t, then the reliability gain G_r for a new configuration χ is the product of π and the number of PMs to be turned off for a duration T_τ: $G_r(\chi) = |y(t) - y(t+1)| \cdot \pi \cdot T_\tau$.

2) ***Impact from wear-and-tear of servers****:* As the start/stop cycle is recognized as the most important factor affecting disk reliability [11], the *annual failure rate* (AFR) with disk start/stop frequency f is empirically quantified as:

$$AFR(f) = \varphi e^{-5} f^2 - \delta e^{-4} f + \vartheta e^{-4} \quad (6)$$

where $\varphi = 3.02, \delta = 2.18$ and $\vartheta = 2.78$ from Google's statistics [11]. As AFR is the ratio of the hours H per year ($8,760$ hours) to the *mean time between failure* (MTBF), and $MTTF \approx MTBF$, the increased AFR can be translated into the decreased MTTF. Then, the decreased MTTF due to disk on-off cycles of a PM s_j can be calculated as: $\triangle MTTF_D^j = H/AFR_j(f) - H/AFR_j(f+1)$.

Since the damage accumulates with each thermal cycle of a processor [13], the MTTF of a processor decreases with the increasing difference in temperature due to on-off thermal cycles: $\triangle MTTF_P \propto (\frac{1}{T_{average} - T_{ambient}})^q$, where $T_{average}$ is the average temperature of a processor, $T_{ambient}$ is the ambient temperature in degrees Kelvin (273.16 plus degrees Celsius), and q is the constant Coffin-Manson exponent, suggested to be 2.35 [13]. Thus the decreased MTTF due to processor on-off thermal cycles of a PM s_j to be turned off in the next time slot, is calculated as Eq. (7), where $T^j_{average\_b}$ and $T^j_{average\_a}$ is the average temperature of the processor before and after consolidation.

$$\triangle MTTF_P^j = [(\frac{T^j_{average\_a} - T_{ambient}}{T^j_{average\_b} - T_{ambient}})^q - 1] \cdot MTTF \quad (7)$$

To reserve reliability, we should limit the number of on-off cycles of PMs. Specifically, we use an array $Count[s_j], \forall s_j \in S$, to record the on-off cycles of each PM, with an upper bound N_{max}. Given that the processors can tolerate more on-off cycles than disks [19], we use the limit of on-off cycles of disks as that of PMs, which is empirically $N_{max} = 24$ according to a latest study [2].

3) ***Impact from temperature rise****:* As some selected PMs will host more workloads after consolidation, their utilization and temperature will increase. We assume that the ambient temperature in a datacenter is constant ($T_{ambient} = 25\,^\circ$C), which is consistent with most real systems. We focus on characterizing the cost of changed temperature due to consolidation. We predict the failure acceleration due to the temperature rise (especially above $50\,^\circ$C [14]), based on the time-to-fail model of Arrhenius equation [20]: $t_f = Ae^{-\triangle E/KT}$, where T denotes the temperature measured in degrees Kelvin at the point when the failure takes place, K is Boltzmann's constant (8.617×10^{-5} in ev/K), A is a constant scaling factor, and $\triangle E$ denotes the activation energy suggested to be 1.25. Thus the acceleration factor (AF) between a higher temperature T_{after} after consolidation and a lower temperature T_{before} before consolidation is: $AF = e^{-\frac{\triangle E}{KT}(\frac{1}{T_{after}} - \frac{1}{T_{before}})}$. Hence, the reduced lifetime ($\triangle MTTF_T$) of a PM s_j with a higher temperature after consolidation is: $\triangle MTTF_T^j = (1 - 1/AF_j) \cdot MTTF$.

Here, an important issue is *how to estimate the temperature of PMs after consolidation in advance*? An intuitive way is to use the server resource utilization to predict its temperature. At this stage, we focus on the temperature impact on processors, which play a major role in server power consumption and temperature. To capture the relationship between the utilization and temperature of a processor, we conduct practical experiments on a Dell PowerEdge 1950 server. We control the server to run the SPECCPU 2006 [21] benchmark, with different utilization levels ranging from 0% to 100%. Meanwhile, we use lm-sensors [22] to record the temperature changes, as

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

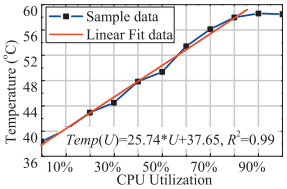


Fig. 1. Relationship between CPU temperature and utilization

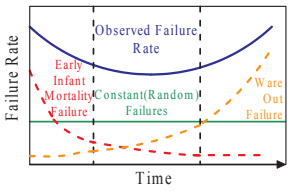


Fig. 2. Bathtub curve of system failure rate over time [23]

depicted in Fig. 1. It shows that when the utilization level is below 80%, there is a *linear* relationship between processor utilization and its temperature. In contrast, when the utilization level exceeds 80%, the temperature is almost constant. We model this as Eq. (8):

$$T(\theta_{CPU}) = \begin{cases} \alpha\theta_{CPU} + \beta, & \theta_{CPU} < 80\% \\ \varrho, & \text{otherwise} \end{cases} \quad (8)$$

Using linear regression techniques, we obtain relevant parameter values: $\alpha = 25.74, \beta = 37.65, \varrho = 58.5$. This implies that when we control the utilization level of servers in an appropriate range, the temperature is likely to be restricted within a threshold T_{max}. For example, T_{max} can be set to 55 °C (where $\theta_{CPU} \approx 80\%$) in our model.

In summary, the total reliability cost C_r due to on-off cycles and increased temperature can be predicted by a product of the decreased MTTF ($\triangle MTTF_D, \triangle MTTF_P$ and $\triangle MTTF_T$) and the reliability utility per unit time π, given as: $C_r(\chi, \triangle MTTF) = \pi \cdot \sum_{j=1}^{|S|} [(h_j(t) - h_j(t+1)) \cdot (\triangle MTTF_P^j + \triangle MTTF_D^j) + h_j(t+1) \cdot \triangle MTTF_T^j]$, where $h_j = 1$ if there is at least one VM located on the PM s_j, otherwise $h_j = 0$. $(h_j(t) - h_j(t+1)) = 1$, if PM s_j will be turned off in the next time slot $t+1$, otherwise 0.

C. The Energy Cost Model

To estimate the energy consumption, we employ the latest empirical non-linear model [10] based on resource utilization: $P(\theta_{CPU}) = (P_{max} - P_{idle}) \cdot (2 \times \theta_{CPU} - \theta_{CPU}^r) + P_{idle}$, where $P(\theta_{CPU})$ is the power consumption when the CPU utilization of a PM is θ_{CPU}, P_{idle} is the basic power consumption when a PM is idle, P_{max} represents the maximum power consumption, and r is a tunable parameter to minimize the square error. Then, the energy consumption $E(\theta_{CPU}^j(t))$ of a PM s_j is the product of $P(\theta_{CPU}^j)$ and the corresponding running duration τ. Let $\rho_\$$ be the electricity charge measured in dollars per kWh, *e.g.,* 0.1$ per kWh. Thus the total saving of electricity cost G_e of a datacenter in the time slot $t+1$ compared to that of the time slot t is: $G_e(\chi, \theta_{CPU}) = \rho_\$ \cdot [\sum_{j=1}^{|S|} E(\theta_{CPU}^j(t)) - \sum_{j=1}^{|S|} E(\theta_{CPU}^j(t+1))]$.

As a PM can be switched off only when all its hosted VMs are migrated to other PMs, it will still consume power P_{idle} during such a transition time T_{off}. Hence, the total switching cost C_{off} for turning off $|y(t) - y(t+1)|$ PMs is: $C_{off}(\chi) = \rho_\$ \cdot |y(t) - y(t+1)| \cdot P_{idle} \cdot T_{off}$.

Besides, the VM migration operation during consolidation also consumes nonnegligible energy, which is observed to increase linearly with the network traffic V_{mig} of VM migration [4]. Accordingly, the energy consumption E_{mig} for migrating VM v_i is $E_{mig}^i = \gamma V_{mig}^i + \eta$, where $\gamma = 0.512, \eta = 20.165$, and V_{mig} is measured in Megabytes as in our previous work [4]. Suppose there are $m(t)$ VMs to be migrated, the total energy cost for VM migration is: $C_{mig}(\chi) = \rho_\$ \cdot \sum_{i=1}^{m(t)} E_{mig}^i$, where $m(t) = \sum_{i=1}^{|V|} \sum_{j=1}^{|S|} (x_{ij}(t) - x_{ij}(t+1))^+$, and the term $x^+ = x$ when $x \geq 0$, otherwise $x^+ = 0$. In summary, the total energy utility of a configuration χ is $U_e(\chi, \theta_{CPU}) = G_e(\chi, \theta_{CPU}) - C_{off}(\chi) - C_{mig}(\chi)$.

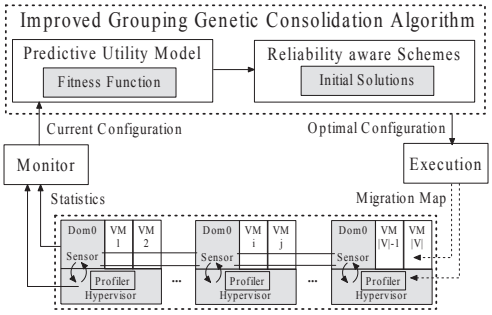


Fig. 3. *RACE*: a reliability-aware and energy-efficient consolidation strategy

D. Optimization Problem of Reliability-Aware Consolidation

Given the utility models above, the goal of *RACE* is to choose a new configuration χ that maximizes the overall utility of a datacenter, which is characterized by the following optimization:

Maximize:

$U(\chi) = \lambda_{SLA} U_{SLA}(\chi) + \lambda_r U_r(\chi, MTTF) + \lambda_e U_e(\chi, \theta_{CPU})$

Subject to:

SLA constraint: $\forall s_j \in S, P_o(s_j) \leq p_{SLA}$

On-off cycles constraint: $\forall s_j \in S, Count[s_j] < N_{max}$

Temperature constraint: $\forall s_j \in S, T_j < T_{max}$

Average load constraint: $\sum_{s_j \in S} Load_j / |S| \geq L_{min}$

Each VM should be placed on a PM: $\forall v_i \in V, \sum_{j=1}^{|S|} x_{ij} = 1$

Number of active PMs: $|S| \geq y(t)$

It is easy to check that the above optimization problem is in general NP-hard, with a prohibitively large space of possible solutions in the order of $O(|V|^{|S|})$. Such a high computational complexity makes it challenging and even impractical to find the global optimal solution. Therefore we resort to a heuristic as explained below.

III. Optimization Approaches for Reliability-Aware Server Consolidation

To efficiently explore the solution space while improving the algorithm convergence rate, *RACE* resorts to a series of reliability-aware resource buffering and VMs-to-PMs remapping heuristics to generate sound initial solutions for an *improved* grouping genetic algorithm (named IG^2CA).

A. Applying an Improved Grouping Genetic Algorithm

Algorithm 1 shows our major procedures in IG^2CA. It begins with a set of initial solutions named *population*, which undergo *crossover* and *mutation* of VMs-to-PMs mapping configurations (χ), and progressively converge to the optimal solution. We use the configuration χ to encode consolidation

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

solutions for its ability in transmitting VMs-to-PMs mapping information from one generation to the next. A *fitness function* is defined as the overall utility $U(\chi)$ (Sec. II-D) of the configuration χ, as it represents the profit of the configuration. Then, the *selection probability* of a solution is defined as the ratio of its individual fitness to the sum of fitness of whole population, given in Eq. (9), where N_P is the population size. The *fitness-based selection* is used to increase the concentration of good groups and avoid the loss of potential solutions in the successive generations. To generate new solutions, two configurations in the population are selected as parents to crossover their subsections of VMs-to-PMs mapping information by swapping or migrating VMs. To avoid premature convergence, each configuration has a mutation probability for slightly modifying the VMs-to-PMs mapping. After N_g times of iteration for crossover and mutation, the fittest solution will be selected as the final near-optimal configuration.

$$P(\chi_i) = F(\chi_i) / \sum_{j=1}^{N_P} F(\chi_j), \quad F(\chi_i) = U(\chi_i) \quad (9)$$

In sum, the complete design and major components of *RACE* is shown in Fig. 3. By periodically monitoring status of VMs and PMs like their resource demands and usages, *RACE* follows our utility models in Sec. II to estimate if it is necessary and profitable for the system to perform consolidation, and then applies the above IG^2CA algorithm to approach optimal configurations with greatest utility for reliability-aware server consolidation.

B. Engineering Feasible Initial Solutions via Reliability-Aware Heuristics

To generate good initial solutions for IG^2CA, we engineer practical reliability-aware heuristics to address the following key challenges, in order to reduce the search space and time.

*1) **How to meet variable demands while alleviating repeated on-off cycles and energy wastage**:* We design a resource buffering heuristic to accommodate workload fluctuations. In addition to packing all VMs with their maximum requirement of resources, each PM reserves a certain amount of safe margin to accommodate potential growth of resource demands from VMs. For example, each PM is allowed to be filled up to 90% of its capacity C_j (*e.g.,* memory capacity), leaving the remaining 10% as a resource buffer. The buffer can absorb the fluctuations of workloads demand. To prevent frequent VM migration and unnecessary on-off cycles, such resource buffer can be linearly extended with the frequency of on-off cycles, described as Eq. (10) for a PM s_j.

$$Buffer_j = C_j \cdot (\psi Count[s_j] + \varsigma)(\%) \quad (10)$$

Empirically, suppose the initial buffer size of each PM is 10% of its capacity, then as the number of on-off cycles reaches the limit $N_{max} = 24$ according to the latest study [2] and our elaboration in Sec. II-B2, the buffer is extended to 20% of its capacity. Accordingly, we can get empirical values of parameters $\psi = 5/12$ and $\varsigma = 10$.

*2) **How many and which PMs can be turned off to save energy without incurring high failure rates of other active PMs**:* We develop an age-load-aware PM selection heuristic based on the "bathtub curve" [23] that shows the relationship between system failure rate and its age. Specifically, the "bathtub curve" consists of three stages: an infant mortality period with a decreasing failure rate, followed by a normal life period with a low and relatively constant failure rate, then ended with a wear-out period with an increasing failure rate. Intuitively, we tend to use PMs in their normal life (empirically preferring 3-year-old PMs), while turning off PMs within the other two life periods. Additionally, we prefer to shut down PMs with light workloads to mitigate VM migration costs. Accordingly, we define a metric called *age-load-quotient* (ALQ) in Eq. (11) to characterize the *Load* (defined in Eq. (5)) and *Age* status of a PM s_j, where $\upsilon = 3$ (years) [23]. *RACE* prefers to turn off the PMs with large values of ALQ, which imply that such PMs have low resource utilization or it is within the infant (or wear-out) age.

$$ALQ_j = (Age_j - \upsilon)^2 / Load_j \quad (11)$$

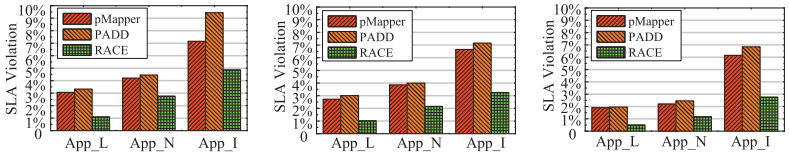
*3) **Where to place VMs from the inactive PMs to balance the temperature of active PMs**:* When performing consolidation, those VMs from the inactive PMs that are turned off will be sorted by their resource demands in a decreasing order, which form a list of migration candidates. Meanwhile, active PMs are sorted by their ALQ in an ascending order to form an ALQ list. For each VM in the migration list, *RACE* attempts to

Algorithm 1: Reliability-aware Heuristics for Generating Feasible Consolidation Solutions.

Input: Current configuration χ_{cur}; multi-dimensional resource utilization and capacity of PMs θ_{jk}, C_{jk}, where k represents the resource types of CPU, memory, disk and network; current statistics of on-off cycles $Count[|S|]$, ages $Age[|S|]$, and $MTTF$ of PMs; w previously measured resource demands.
Output: New configuration χ_{new}; overall utility $U(\chi_{new})$; VM migration map MigMap (which VMs assigned to which PMs).

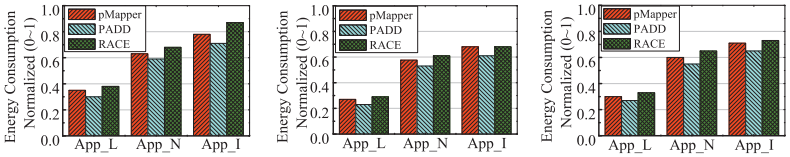
```
χ_new = χ_cur;
Compute average Load of a datacenter using Eq. (5);
if average Load < L_min then
    Compute the ALQ of PMs using Eq. (11);
    Sort all PMs in S by ALQ in an ascending order;
    for s_j ∈ S do
        if ALQ(s_j) > λ_ALQ (Threshold) then
            //NodeOff is a list of candidate PMs to be turned off;
            NodeOff ⟵ s_j; S = S − NodeOff;
    Sort the VMs from the PMs in NodeOff by their resource
    demands in a decreasing order, stored in a VMOff list;
    for v_i ∈ VMOff do
        for s_t ∈ S do
            Predict s_t's temperature T_{s_t} and its resource buffer
            Buffer_{s_t} using Eq. (8) and Eq. (10);
            if (Count[s_t] < N_max) and (T_{s_t} < T_max) and
            ((Buffer_{s_t} + Σ_{v_i ∈ s_t} R_i) < C_t) then
                FeasiblePMs ⟵ s_t;
        if FeasibleServers ≠ φ then
            //Find target PM s_o with the minimized estimated
            temperature difference for hosting v_i;
            s_o ⟵ FindTargetServer(FeasibleServers);
            //Migrate v_i to s_o, and update χ_new;
            MigMap.add(v_i, s_o); x_ij^new = 0; x_io^new = 1;
        else
            break; //Stop re-mapping process;
    Calculate the overall utility according to Eq. (1);
    if U(χ_new) < 0 then
        Reject the configuration, the datacenter remains unchanged;
    else
        return χ_new, MigMap, U(χ_new).
```


978-1-4673-4510-1/12/$31.00 ©2012 IEEE



(a) Small VM instances (b) Large VM instances (c) Extra-large VM instances

Fig. 4. Comparison of SLA violations in term of the percentage of mismatch between demand for and supply of PM resources



(a) Small VM instances (b) Large VM instances (c) Extra-large VM instances

Fig. 5. Comparison of energy consumption under consolidation normalized by that of the base case without consolidation

find an appropriate active PM in ALQ list to host the VM. For all feasible PMs, *RACE* estimates the temperature difference before and after hosting this VM on those candidate PMs, based on our modeling analysis in Sec. II-B. It then migrates the VM to a PM that suffers from the minimized estimated temperature difference.

Algorithm 1 gives the complete pseudo-code of the above reliability-aware heuristics for generating good initial solutions as input to IG^2CA. When the average $Load$ of a datacenter is below L_{min}, based on our utility models in Sec. II, *RACE* will make decisions of whether and how to perform consolidation.

IV. Performance Evaluation

A. Experimental Setup

We simulate a datacenter based on an open-source discrete-event simulator CloudSim [24]. It can offer us a repeatable and controllable environment. Specifically, we simulate 100 PMs with $P_{idle} = 185$W and $P_{max} = 300$W, each of which has quad-core processors with $2,000$, $2,500$, $3,000$ or $3,500$ million instructions per second (in accordance with the Amazon EC2 cloud service), 16 GB memory, 1 TB storage and 10 Gb/s network bandwidth. We configure three representative types of VM instances like Amazon EC2, for comparing different consolidation strategies, including *small*, *large*, and *extra-large* VM instances described in Table. I.

TABLE I
Configuration of representative types of VM instances

Type	CPU (core)	Memory (GB)	Storage (GB)
Small	1	1.7	160
Large	2	7.5	850
Extra-Large	4	15	1,690

A datacenter broker generates user requests in a Poisson process with an arrival rate R_{in}. We emulate *light*, *normal*, and *intensive* application workloads, denoted as App_L, App_N, and App_I, respectively, with different arrival rates $R_{in} = 5, R_{in} = 15$ and $R_{in} = 30$ requests per second. The *request size* is defined as the processing time of the request, following a bounded random distribution. The maximum request size is set to 5 time intervals ($5 \times T_{timeout}$ in Sec. II-A), while the minimum request size is 0.05 time intervals. We evaluate *RACE* against two representative server consolidation strategies: *pMapper* [5] and *PADD* [9]. We also consider the *base case* without consolidation as a performance baseline.

B. Evaluation of Performance SLAs and Energy Efficiency

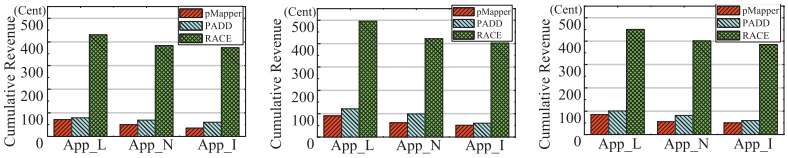
First, we compare the performance SLAs under *RACE* against *pMapper* and *PADD* strategies, by running different levels of application workloads App_L, App_N, and App_I for 30 time intervals on three representative VM instances in Table. I. We then calculate the SLA violations of different consolidation strategies, based on the logs of requested resources of each VM and the allocated resources from each PM. As shown in Fig. 4, server consolidation can indeed incur increasing SLA violations, as application workloads become more intensive. This is due to the resource shortage for accommodating workload fluctuations. Benefiting from the resource buffering mechanism in Sec. III-B, *RACE* outperforms *pMapper* and *PADD* in mitigating SLA violations for all three types of VM instances. As both *pMapper* and *PADD* strategies tend to catch every consolidation opportunity without considering the costs of PM on-off cycles and VM migrations, the energy saved by them may be outweighed by increasing penalty of SLA violations.

Next, Fig. 5 compares the energy consumption of *RACE* against *pMapper* and *PADD* strategies, which is normalized as the ratio of their energy consumption to that of the base case without consolidation. We observe that all three types of consolidation strategies can save energy costs compared to the base case, especially for light and normal application workloads. While *PADD* and *pMapper* place more emphasis on minimizing energy consumption, *RACE* seeks to achieve a balanced tradeoff between its slightly higher energy consumption and its reliability gain for alleviating the impacts from on-off cycles and temperature rise of PMs. This reflects the intention of our multi-objective optimization for overall system utility, and will be further evidenced in the next subsection.

C. Comparison of Reliability-Aware Profitability

Further, we compare the cumulative overall utility (Eq. (1)) of three types of consolidation strategies, in term of U.S. cent over time under different types of VM instances and application workloads. Fig. 6 shows that *RACE* outperforms

978-1-4673-4510-1/12/$31.00 ©2012 IEEE



(a) Small VM instances (b) Large VM instances (c) Extra-large VM instances

Fig. 6. Comparison of cumulative utility in U.S. cent under different types of VM instances and application workloads

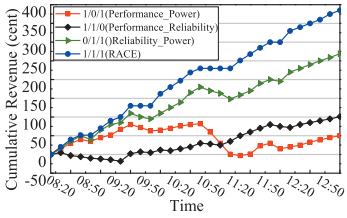


Fig. 7. Comparison of cumulative utility over time, under different settings of respective utility weights λ_{SLA}, λ_r and λ_e

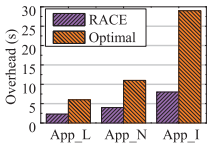


Fig. 8. Overhead of *RACE* in term of additional execution time of application workloads

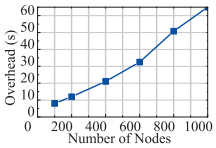


Fig. 9. Overhead of *RACE* increases linearly with the scale of a datacenter

pMapper and *PADD* significantly, by avoiding unprofitable and aggressive reconfigurations. In contrast, the aforementioned SLA penalty under *pMapper* and *PADD* strategies together with their reliability costs — due to repeated on-off thermal cycles, wear-and-tear and temperature rise of PMs — are more likely to outweigh their energy savings, especially during time intervals with highly fluctuating demands from VMs. Recall that *PADD* can save more energy (Fig. 5) while maintaining similar performance levels as *pMapper* (Fig. 4), and hence the former achieves higher cumulative profit than the latter does for most cases. We also observe that with less intensive workloads and large VM instances, it is more suitable for a datacenter to improve its overall profit through consolidation.

Then, we proceed to take a closer look at the effects of different settings of respective utility weights λ_{SLA}, λ_r and λ_e in our multi-objective utility model in Sec. II. Essentially, they represent the design preferences of a datacenter over different combinations of performance SLA, reliability and energy concerns. Specifically, we examine four representative cases, including *RACE* with $\lambda_{SLA}/\lambda_r/\lambda_e = 1/1/1$, compared with $\lambda_{SLA}/\lambda_r/\lambda_e = 0/1/1$, $\lambda_{SLA}/\lambda_r/\lambda_e = 1/0/1$ and $\lambda_{SLA}/\lambda_r/\lambda_e = 1/1/0$, each of which adjusts the tradeoff among performance, reliability and energy objectives. For each of these settings, we emulate a datacenter that runs normal application workloads App_N on small VM instances for 30 time intervals. Fig. 7 compares the cumulative overall utility over time under different settings. We observe that *RACE* achieves increasingly higher overall utility over time than the other settings. The rationale is that *RACE* controls energy-efficient server consolidation with mitigated penalties from reliability impacts and SLAs violations, so as to maintain a holistic balance between the overall costs and benefits of consolidation. Interestingly, the cross between curves with $\lambda_{SLA}/\lambda_r/\lambda_e = 1/1/0$ and $\lambda_{SLA}/\lambda_r/\lambda_e = 1/0/1$, respectively, clearly captures the tradeoff between *the long-term reliability-aware profit racing* against *the short-term energy savings*. Due to over-aggressive energy savings that potentially incur higher SLA penalty and reliability cost, traditional consolidation strategies solely focusing on the performance-energy tradeoff may lead to the lowest profit for datacenters eventually. Here, we only uses "binary" weights, but values between 0 and 1 is permitted. We will analysis the sensitivity to inputs in future work.

D. Validation of Overhead and Scalability

Finally, we validate the overhead and scalability of *RACE*. Specifically, by running the three types of application workloads App_L, App_N, and App_I on small VM instances. We record the time interval between their start and end time as the execution time when using *RACE*. This is compared with an *optimal* case by setting the number of iteration times N_g of IG^2CA in Sec. III to infinity, which keeps searching the best solution until no greater utility can be brought forth. The overhead of using *RACE* and optimal case is defined as the additional execution time of application workloads compared to that without using them, as shown in Fig. 8. Benefiting from our reliability-aware heuristics for generating good initial solutions in Sec. III-B and fine-tuned parameter settings in Sec. IV-A, *RACE* incurs remarkably less overhead than the optimal case involving an exhaustive search, which leads to an exponentially increasing execution time as the workloads become more intensive.

As characterized by the utility optimization problem in Sec. II-D, the solution space grows exponentially with the number of PMs and VMs. To improve the scalability of *RACE*, it can be implemented in a distributed manner, with multi-level hierarchical controls. Specifically, the simulated datacenter is managed by several local controllers for different subsets of PMs in the first level, while a global controller manages the whole system. The local controllers perform consolidation of respective PM subsets in a fine granularity (*e.g.,* every few minutes) based on Algorithm 1, while the global controller applies IG^2CA to reconfigure the whole system less frequently (*e.g.,* every dozens of minutes or hourly). The execution time of workloads is shown in Fig. 9. We observe that *RACE* takes just about one minute to consolidate workloads on $1,000$ PMs. As *RACE* can reduce the search space via reliability-aware heuristics and a distributed implementation, the execution time of using *RACE* can be approximately maintained as a linear relationship with the number of PMs. This implies the viability of *RACE* to be deployed in large-scale datacenters with acceptable overhead.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE

V. Related Work

Dynamic server consolidation via VM migration techniques has been extensively studied to right-size the number of active servers in datacenters for power-saving purposes. For instance, a prior solution called PADD [9] used a two-level adaptive buffering scheme to reduce overall system energy consumption while avoiding SLA violations. While we are inspired by this work to incorporate a resource buffering heuristic to alleviate the impact of frequent on-off cycles under workload variations, our model and strategy in *RACE* further take into account VM migration costs that were often ideally simplified in existing studies. Though the pMapper strategy [5] jointly modeled VM migration overhead, power and performance benefits of power-aware application placement, it mainly operated in a static and offline manner. In contrast, *RACE* performs reliability-aware consolidation using a dynamic and online strategy. A more sophisticated consolidation framework called Entropy [3] explored sequences of migrations to seek for a globally optimal reconfiguration, yet along with increasing time complexity.

There also exist several studies on understanding various factors and bottlenecks that adversely affect server consolidation. Verma *et al.* [7] found that the performance isolation and virtualization overhead with multiple VMs could become key bottlenecks for server consolidation [8]. A more recent study [6] showed that consolidation of workloads with dissimilar characteristics can reduce the total resource requirements and power consumption significantly. However, it solely focused on the performance-energy tradeoff, without attention on the impacts of wear-and-tear of servers.

In addition, as repeated on-off cycles can increase the wear-and-tear of server components [11], such factors have also been considered by recent server provisioning mechanisms. Guenter *et al.* [2] have applied a Markov state model to dynamically provision servers to meet workload demands, while minimizing the reliability cost due to repeated on-off cycles. More recently, Lin *et al.* [17] modeled not only the delay and energy costs of active servers, but also the switching costs of energy and wear-and-tear. We further consider impact of temperature rise and engineer practical reliability-aware resource buffering and VMs-to-PMs re-mapping heuristics to control reliability during power-aware VM consolidation.

VI. Conclusion

This paper presented *RACE*, a reliability-aware server consolidation strategy that optimizes *overall and long-term* utility of virtualized datacenters, by particularly focusing on (1) the reliability and lifetime costs of servers due to repeated on-off thermal cycles and temperature rise, (2) the costs for migrating VMs, as well as (3) potential violations of performance SLAs — *in a holistic multi-objective utility model*. An improved grouping genetic algorithm has been proposed to solve this optimization problem, based on sound initial solutions constructed by our reliability-aware resource buffering and VMs-to-PMs re-mapping heuristics. Extensive experiments have demonstrated that *RACE* can avoid unprofitable reconfigurations under dynamic server consolidation in the long term, and thus outperform *pMapper* and *PADD* that solely focused on the performance-energy tradeoff in an aggressive manner. *RACE* is able to be deployed in large-scale datacenters with acceptable overhead.

VII. Acknowledgement

The research was supported in part by National Natural Science Foundation of China (NSFC) under grants No.61133006 and No.61103176, and by research fund of young scholars for the Doctoral Program of Higher Education, Ministry of Education, China under grant No.20110142120079.

References

[1] A. Qureshi, "Power-Demand Routing in Massive Geo-Distributed Systems," Ph.D. dissertation, Massachusetts Institute of Technology, 2010.
[2] B. Guenter, N. Jain, and C. Williams, "Managing Cost, Performance, and Reliability Tradeoffs for Energy-Aware Server Provisioning," in *Proc. of IEEE INFOCOM*, Apr. 2011.
[3] F. Hermenier, X. Lorca, and J.-M. Menaud, "Entropy: A Consolidation Manager for Clusters," in *Proc. of VEE*, Mar. 2009.
[4] H. Liu, C. Z. Xu, and H. Jin, "Performance and Energy Modeling for Live Migration of Virtual Machines," in *Proc. of HPDC*, June 2011.
[5] A. Verma, P. Ahuja, and A. Neogi, "PMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," in *Proc. of ACM Middleware*, Dec. 2008.
[6] Q. Zhu and J. Zhu, "Power-Aware Consolidation of Scientific Workflows in Virtualized Environments," in *Proc. of SC*, Nov. 2010.
[7] A. Verma, G. Dasgupta, T. Nayak, P. De, and R. Kothari, "Server Workload Analysis for Power Minimization Using Consolidation," in *Proc. of USENIX ATC*, June 2009.
[8] A. Verma, P. Ahuja, and A. Neogi, "Power-aware Dynamic Placement of HPC Applications," in *Proc. of ICS*, Oct. 2008.
[9] M. Y. Lim, F. Rawson, T. Bletsch, and V. W. Freeh, "PADD: Power Aware Domain Distribution," in *Proc. of IEEE ICDCS*, June 2009.
[10] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu, "Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures," in *Proc. of ICDCS*, June 2010.
[11] T. Xie and Y. Sun, "Sacrificing Reliability for Energy Saving: Is It Worthwhile for Disk Arrays?" in *Proc. of IPDPS*, Apr. 2008.
[12] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "Characterizing Processor Thermal Behavior," in *Proc. of ASPLOS*, Mar. 2010.
[13] J. Srinivasan, S. Adve, and P. Bose, "Lifetime Reliability: Toward an Architectural Solution," in *Proc. of IEEE Micro*, Jan. 2005.
[14] N. El-Sayed, I. Stefanovici, G. Amvrosiadis, A. Hwang, and B. Schroeder, "Temperature Management in Data Centers: Why Some (Might) Like It Hot," in *Proc. of SIGMETRICS*, June 2012.
[15] K. Vishwanath and N. Nagappan, "Characterizing Cloud Computing Hardware Reliability," in *Proc. of ACM SoCC*, June 2010.
[16] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. Maltz, and I. Stoica, "Surviving Failures in Bandwidth-Constrained Datacenters," in *Proc. of SIGCOMM*, Aug. 2012.
[17] M. Lin, A. Wierman, L. H. Andrew, and E. Thereska, "Dynamic Right-sizing for Power-proportional Data Centers," in *Proc. of IEEE INFOCOM*, Apr. 2011.
[18] D. S. Stoffer and R. H. Shumway, *Time Series Analysis and Its Applications*. New York: Springer Verlag, 2000.
[19] Berkeley, "A Case for Adaptive Datacenters to Conserve Energy and Improve Reliability." University of California, Berkeley Reliable Adaptive Distributed (RAD) Systems Laboratory, 2008.
[20] P. W. Hale, "Acceleration and Time to Fail," *Quality and Reliability Engineering International*, vol. 2, no. 4, pp. 259–262, 1986.
[21] SPEC CPU2006. [Online]. Available: http://www.spec.org/cpu2006/
[22] http://www.lm-sensors.org.
[23] http://www.weibull.com/hotwire/issue21/hottopics21.html.
[24] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. 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, Wiley Press*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

978-1-4673-4510-1/12/$31.00 ©2012 IEEE