Reliability-aware Server Consolidation for Balancing Energy-Lifetime Tradeoff in Virtualized Cloud Datacenters

Wei Deng, Fangming Liu, Hai Jin∗†, Xiaofei Liao, Haikun Liu

School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China.
Email: {wdeng, fmliu, hjin, xfliao}@hust.edu.cn

SUMMARY
Server consolidation using virtualization technologies allow large-scale datacenters to improve resource utilization and energy efficiency. However, most existing consolidation strategies solely focused on balancing the tradeoff between performance 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, ware-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. Copyright © 2013 John Wiley & Sons, Ltd.

1. INTRODUCTION
It has been widely acknowledged that energy costs become skyrocketing with the recent expanding of large-scale datacenters. In 2011, commercial datacenters in U.S. consumed more than 120 billion kWh at a cost of $9 billion dollars [1], such as Google with over 1,120GWh and $67M, and Microsoft with over 600GWh and $36M [2]. However, a tremendous amount of such energy is actually over-provisioned for accommodating fluctuating application workloads and peak user demands. Reportedly, many servers in datacenters operate between merely 10 and 50 percent of their maximum utilization levels for most of their runtime [3].

Fortunately, dynamic workload consolidation among different servers based on virtualization technologies [4] 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 [5], 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 [4–7], resource contention and performance interference between co-located VMs [8–10], as well as cloud SLA violations [11, 12].

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 [13] has not yet been well understood and alleviated. As illustrated by Fig. 1, 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, ware-and-tear, and temperature rise:

First, due to workload fluctuations in datacenters, a certain set of servers like PM 1 in Fig. 1 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 [14, 15]. Furthermore, the on-off thermal cycle of processor contributes to another important factor causing fatigue failures [16, 17]. 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 [1, 18].

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, such as PM 2 and PM 3 in Fig. 1. Both the utilization and temperature of such servers will increase, which may affect their lifetime. Reportedly, each 10°C temperature rise would reduce the server component life by 50% [19]. Temperature is an important influential factor in the reliability, performance and power consumption of modern processors and servers in datacenters [16, 20]. Additionally, software often exhibits an increasing failure rate over time, typically because of increasing and unbounded resource consumption [21]. Since the certain selected PMs will host more workloads after consolidation and run longer time, the operating systems and virtual machine hypervisors of these PMs will have larger failure rates.

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 [1]. These failures can incur partial or complete outage of services that cost about $5,000 per minute [22]. Although Xie and Sun provided a reliability model for power-aware disk system [14], and recent studies considered the impacts of on-off cycles for power-aware server provisioning [1, 13, 23], 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 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 named RACE, based on our previous work [24]. It focuses on characterizing and analyzing the above problem.
as a multi-objective optimization, based on an utility model that we designed 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 re-mapping heuristics for generating good initial solutions and improving the convergence rate. We validate the effectiveness, scalability and overhead of RACE through extensive simulation experiments using the open-source CloudSim tool [25]. The results show that our strategy can improve the overall utility of datacenters while avoiding unprofitable consolidation in the long term, compared with pMapper [7] and PADD [11] strategies for server consolidation.

The rest of the paper is organized as follows. We discuss the related works in Sec. 2. Sec. 3 constructs our utility models and the multi-objective optimization problem for reliability-aware server consolidation. Sec. 4 presents our improved grouping genetic algorithm to efficiently solve the problem. Extensive simulation results are demonstrated in Sec. 5. Then, we discuss the limits and our future work in Sec. 6. Finally, we conclude this paper in Sec. 7.

2. RELATED WORK

In this section, we discuss the research most pertinent to this work as follows.

2.1. Virtual machine consolidation

Dynamic server consolidation via VM migration techniques has been extensively studied to rights-size the number of active servers in datacenters for power-saving purposes. For instance, a prior solution called PADD [11] 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 [7] 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 server consolidation using a dynamic and online strategy. Another more sophisticated consolidation framework called Entropy [4] explored the sequences of migrations to seek for a globally optimal reconfiguration (with the minimized number of servers), yet along with increasing time complexity. Mistral [13] controls the adaptive cost due to VM migration when adjusting datacenter capacity, while we model the reliability cost due to on-off cycles and temperature aggregation.

There also exist several studies on understanding various factors and bottlenecks that adversely affect server consolidation. Verma et al. [9] found that the performance isolation and virtualization overhead with multiple VMs could become key bottlenecks for server consolidation. In particular, the working set size in term of the amount of memory needed was observed as a critical factor for determining the placement of applications across servers [10]. A more recent study [8] 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 during consolidation, without adequate attention on the impacts of reliability and wear-and-tear of servers — which is the focus of our study.

2.2. Reliability aware power management

Repeated on-off cycles can increase the wear-and-tear of server. When managing server operational costs, Chen et al. [26] modeled the turning on cost of servers, while Qian et al. [18] modeled both
the turning on and off costs of servers. However, SLA guarantee and impact of temperature rise are not discussed in their work. For dynamically provisioning servers, Guenter et al. [1] had applied a Markov state model to meet workload demands, while minimizing the reliability cost due to repeated on-off cycles. Lin et al. [23] modeled not only the delay and energy costs of active servers, but also the switching costs of energy and wear-and-tear for toggling servers into and out of power-saving modes. However, these work haven’t consider the impact of temperature rise and migration cost of VMs specified in virtualized datacenters. There are other different groups of works with different objectives such as power management in wireless sensor networks [27–29], and minimization of power costs and carbon emission in cloud datacenters [30–33].

In addition, temperature impacts reliability and power consumption of servers components and internet datacenters. Many works have characterized the reliability impacts. For example, Xie et al. [14] presented a reliability model related to operating temperature, utilization, and disk speed transition frequency. Based on processor temperature and utilization, Ramp [17] used ware out models to calculate the MTTF. Jayanth et al. [16] showed that thermal cycling such as power on and off and frequency changes affect the lifetime of the processor. Besides, temperature has a linear relationship with various timing integrity but exponential impact on leakage power. Similarly, for large cloud computing infrastructures, Vishwanath et al. [20] characterized server repair/failure rates and find that location of the datacenter and the manufacturer are the dominant indicators of failures, but not age, configuration etc. Additionally, several works have proposed thermal-aware and temperature-aware server provisioning and workload distribution approaches [34–36]. However, these works provided understanding on performance impact and cooling efficiency impact from temperature, but little attention was paid to the reliability impact from temperature and load-induced impact due to consolidation. We further engineer practical reliability-aware resource buffering and VMs-to-PMs re-mapping heuristics within our RACE strategy, so as to control on-off cycles and temperature impacts during power-aware server consolidation.

3. RACE: 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 \( \xi \) (e.g., in a timescale of hours) is slotted into \( T \) equal time slots, where the key system parameters are given in Table 1. 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 \( \chi \), \( x_{ij} = 1 \) if PM \( s_j \) hosts VM \( v_i \), otherwise \( x_{ij} = 0 \). In response to dynamic workloads, it is critical to decide when the consolidation should be performed and how it can be made 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, ware-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 \( \chi \) for the next time slot can be expressed as a weighed combination:

\[
U(\chi) = \lambda_{SLA} U_{SLA}(\chi) + \lambda_r U_r(\chi) + \lambda_e U_e(\chi),
\]

where \( \lambda_{SLA} \), \( \lambda_r \), and \( \lambda_e \) 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 \( \chi \) 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

\[\text{Int. J. Commun. Syst. (2013)}\]

Copyright © 2013 John Wiley & Sons, Ltd. DOI: 10.1002/dac
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?

### 3.1. 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):

$$P_o = \text{Probability}\left[\sum_{i=1}^{V} x_{ij} R_{ik}(t) \geq C_{jk} \leq p_{SLA}\right],$$

$$\forall v_i \in V, \forall s_j \in S, \forall t \in T, k = 0, 1, 2, 3,$$

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 $\chi$, the SLA utility of datacenter is:
\[
U_{SLA}(\chi) = \sum_{i=1}^{|S|} U_{SLA}(s_i),
\]
where $U_{SLA}(s_i) = \begin{cases} R_{SLA}(i), & \text{if } P_o \leq p_{SLA}, \\ P_{SLA}(i), & \text{otherwise}. \end{cases}$

Here, we have two types of penalty models: (1) Fixed Penalty: The fixed penalty $P_{SLA}$ is charged whenever the cloud service provider fails to fulfill current resource demand. (2) Proportional Penalty: The penalty $P_{SLA}(i)$ is proportional to the difference between the demand of a VM $R_i$ and the provisioned resource $RP_i$. Let $P_{SLA}^{max}$ be the maximum penalty for no resource provisioned. Then, the penalty is computed as: $P_{SLA}(i) = P_{SLA}^{max}(R_i - RP_i)$. On the other hand, we assume that the monetary reward $R_{SLA}(i)$ is constant whenever the cloud service provider fulfils current capacity demand.

Note that, though this formulation does ignore resource contention and performance interference between co-located VMs, such issues are beyond the scope of this paper; nevertheless, our models merge nicely with proposals such as [8, 9] for these goals. 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) [37] prediction technique. It predicts the workload demands $R_{ik}(t+1)$ of the next time slot based on the average of the $w$ previously measured demands $R_{ik}^m$ via the following equation:
\[
R_{ik}(t+1) = (1 - \lambda) \cdot R_{ik}^m(t) + \lambda \cdot \frac{1}{w} \sum_{l=1}^w R_{ik}^m(t-l),
\]
where $\lambda$ is used to weigh the current value against past historical measurements. As we will use in our experiments in Sec. 5, a typical setting is $w = 3$ and $\lambda = 0.5$ [13] to give balanced weights to the current and historical measurements.

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 utilisations of a PM as the product of its CPU, memory, disk and network utilization ratios:
\[
Load = \frac{1}{(1 - \theta_{CPU}) \cdot (1 - \theta_{Mem}) \cdot (1 - \theta_{Disk}) \cdot (1 - \theta_{Net})},
\]
where $\theta_{CPU}$, $\theta_{Mem}$, $\theta_{Disk}$ and $\theta_{Net}$ are the corresponding average utilization ratios ($0 \sim 100\%$) of the PM resources at each time slot. Such resource utilization will be set to $1 - \varepsilon$ ($\varepsilon \rightarrow 0$) 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 [23] in practice.

3.2. 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 impact from increased load of active PMs $C_{load}$, 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) $\Delta{MTTF}$ to evaluate the reliability costs. In particular, temperature is the dominant factor in reliability, leakage power consumption, and performance of modern processors [16], and 70% of server failures is due to disks [20]. Therefore, 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 [38], which is quantified as the reliability gain $G_r$ in next subsection. Then, the overall reliability utility of a configuration $\chi$ is calculated as $U_r(\chi, MTTF) = G_r(\chi) - C_r(\chi, \Delta{MTTF}) - C_{load}$. 

3.2.1. Utility of reliability gain: We denote the reliability utility per unit time as \( \pi \), 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, \( \pi \) 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 \( \pi \) 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 \( \chi \) is the product of \( \pi \) and the number of PMs to be turned off for a duration \( T_r \): 
\[
G_r(\chi) = |y(t) - y(t+1)| \cdot \pi \cdot T_r.
\]

3.2.2. Impact from wear-and-tear of servers: As the start/stop cycle is recognized as the most important factor affecting disk reliability [14], the annual failure rate (AFR) with disk start/stop frequency \( f \) is empirically quantified as:
\[
AFR(f) = \varphi e^{-5 f^2} - \delta e^{-4 f} + \theta e^{-4},
\]
where \( \varphi = 3.02, \delta = 2.18 \) and \( \theta = 2.78 \) from Google’s statistics [14]. 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:
\[
\Delta MTTF^j_P = H \frac{AFR_j(f)}{AFR_j(f + 1)}.
\]
Since the damage accumulates with each thermal cycle of a processor [17], the MTTF of a processor decreases with the increasing difference in temperature due to on-off thermal cycles: 
\[
\Delta MTTF^j_P \propto \left( \frac{1}{T_{\text{average},a} - T_{\text{ambient}}} \right)^q,
\]
where \( T_{\text{average},a} \) is the average temperature of a processor, \( T_{\text{ambient}} \) 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 [17]. 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. (8), where \( T_{\text{ambient}, b}^j \) and \( T_{\text{ambient}, a}^j \) is the average temperature of the processor before and after consolidation:
\[
\Delta MTTF^j_P = \left( \left( \frac{T_{\text{average}, b}^j}{T_{\text{average}, a}^j} - T_{\text{ambient}}^j \right)^q - 1 \right) \cdot MTTF.
\]
To reserve reliability, we should limit the number of on-off cycles of PMs. Specifically, we use an array \( \text{Count}[s_j], \forall s_j \in S \), to record the on-off cycles of each PM, with an upper bound \( N_{\text{max}} \). Given that the processors can tolerate more on-off cycles than disks [38], we use the limit of on-off cycles of disks as that of PMs, which is empirically \( N_{\text{max}} = 24 \) according to a latest study [1].

3.2.3. 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 (\( T_{\text{ambient}} = 25^\circ C \)), 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, based on the time-to-fail model of Arrhenius equation [39]: 
\[
t_f = Ae^{-\Delta E/KT},
\]
where \( T \) denotes the temperature measured in degrees Kelvin at the point when the failure takes place, \( K \) is Boltzmann’s constant (\( 8.617 \times 10^{-5} \) in eV/K), \( A \) is a constant scaling factor, and \( \Delta E \) denotes the activation energy suggested to be 1.25. Thus the acceleration factor (AF) between a higher temperature \( T_{\text{after}} \) after consolidation and a lower temperature \( T_{\text{before}} \) before consolidation is: 
\[
AF = e^{\frac{\Delta E}{K} \left( \frac{T_{\text{after}}}{T_{\text{before}}} - 1 \right)}.
\]
Hence, the reduced lifetime \( \Delta MTTF_P^j \) of a PM \( s_j \) with a higher temperature after consolidation is:
\[
\Delta MTTF^j_P = \left( 1 - \frac{1}{AF_j} \right) \cdot MTTF.
\]
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 [40] benchmark, with different utilization levels ranging from 0% to 100%. Meanwhile, we use lm-sensors [41] to record the temperature changes, as depicted in Fig. 2. 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. (10):

\[
T(\theta_{CPU}) = \begin{cases} 
\alpha \theta_{CPU} + \beta, & \theta_{CPU} < 80\%, \\
\phi, & \text{otherwise.}
\end{cases}
\]  

(10)

Using linear regression techniques, we obtain relevant parameter values: \(\alpha = 25.74, \beta = 37.65, \phi = 58.5\). 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 (\(\Delta MTTF_D, \Delta MTTF_P\) and \(\Delta MTTF_T\)) and the reliability utility per unit time \(\pi\), given as: \(C_r(\chi, \Delta MTTF) = \pi \cdot \sum_{j=1}^{[\chi]} (h_j(t) - h_j(t+1)) \cdot (\Delta MTTF_D^j + \Delta MTTF_P^j + h_j(t+1) \cdot \Delta MTTF_T^j)\), where \(h_j = 1\) if there is at least one VM located on the PM \(s_j\), otherwise \(h_j = 0\). And \((h_j(t) - h_j(t+1)) = 1\), if PM \(s_j\) will be turned off in the next time slot \(t+1\), otherwise \(0\).

3.2.4. Impact from load induced failures: Software often exhibits an increasing failure rate over time, typically because of increasing and unbounded resource consumption [21]. The state of the software degrades gradually with time, inevitably resulting in undesirable consequences. Since the certain selected PMs will host more workloads after consolidation and run longer time, the operating systems and virtual machine hypervisors of these PMs will have larger failure rates. We model the impact of load induced failures for PM \(s_j\) as \(C_{load}^j\), which is a non-decreasing function of the density of workload and running time of the PM, i.e., \(C_{load}^j(t) = f(\text{Load}_j(t), T_j(t))\), where \(\text{Load}_j(t)\) is the multi-dimensional resource utilizations of \(s_j\) at time slot \(t\), and \(T_j(t)\) is the running time of \(s_j\) since it reboots last time until \(t\). Then, for all of the PMs, the cost due to increased load is \(C_{load} = \sum_{j=1}^{[\chi]} h_j(t+1)C_{load}^j(t+1)\).

3.3. The Energy Cost Model

To estimate the energy consumption, we employ the latest empirical non-linear model [13] based on resource utilization:

\[
P(\theta_{CPU}) = (P_{max} - P_{idle}) \cdot (2 \times \theta_{CPU} - \theta_{CPU}^2) + P_{idle},
\]  

(11)
where \( P(\theta_{\text{CPU}}) \) is the power consumption when the CPU utilization of a PM is \( \theta_{\text{CPU}} \), \( P_{\text{idle}} \) is the basic power consumption when a PM is idle, \( P_{\text{max}} \) represents the maximum power consumption, and \( r \) is a tunable parameter to minimize the square error. Then, the energy consumption \( E(\theta_{\text{CPU}}(t)) \) of a PM \( s_j \) is the product of \( P(\theta_{\text{CPU}}) \) and the corresponding running duration \( \tau \). Let \( p_e \) be the electricity charge measured in dollars per kWh, e.g., $0.18$ 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:

\[
G_e(\chi, \theta_{\text{CPU}}) = p_e \cdot \left[ \sum_{j=1}^{\lvert S \rvert} E(\theta_{\text{CPU}}^j(t)) - \sum_{j=1}^{\lvert S \rvert} E(\theta_{\text{CPU}}^j(t+1)) \right]. \tag{12}
\]

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

\[
C_{\text{off}}(\chi) = p_e \cdot |y(t) - y(t+1)| \cdot P_{\text{idle}} \cdot T_{\text{off}}. \tag{13}
\]

Besides, the VM migration operation during consolidation also consumes nonnegligible energy, which is observed to increase linearly with the network traffic \( V_{\text{mig}} \) of VM migration \cite{5}. Accordingly, the energy consumption \( E_{\text{mig}} \) for migrating VM \( v_i \) is \( E_{\text{mig}} = \gamma V_{\text{mig}} + \eta \), where \( \gamma = 0.512, \eta = 20.165 \) and \( V_{\text{mig}} \) is measured in Megabytes as in our previous work \cite{5}. Suppose there are \( m(t) \) VMs to be migrated, the total energy cost for VM migration is:

\[
C_{\text{mig}}(\chi) = p_e \cdot \sum_{i=1}^{m(t)} E_{\text{mig}}^{i}, m(t) = \sum_{i=1}^{\lvert V \rvert} \sum_{j=1}^{\lvert S \rvert} (x_{ij}(t) - x_{ij}(t+1))^+. \tag{14}
\]

The term \( x^+ = x \) when \( x \geq 0 \), otherwise \( x^+ = 0 \). In summary, the total energy utility of a configuration \( \chi \) is \( U_e(\chi, \theta_{\text{CPU}}) = G_e(\chi, \theta_{\text{CPU}}) - C_{\text{off}}(\chi) - C_{\text{mig}}(\chi) \).

### 3.4. The Optimization Problem of Reliability-Aware Server Consolidation

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

\[
\max \quad U(\chi) = \lambda_{\text{SLA}} U_{\text{SLA}}(\chi) + \lambda_{\tau} U_{\tau}(\chi, MTTF) + \lambda_{e} U_{e}(\chi, \theta_{\text{CPU}})
\]

\[
\text{s.t.} \quad \text{SLA constraint : } \forall s_j \in S, P_o(s_j) \leq p_{\text{SLA}},
\]

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

\[
\text{Temperature constraint : } \forall s_j \in S, T_j < T_{\text{max}},
\]

\[
\text{Average load constraint : } \sum_{s_j \in S} \text{Load}_j/|S| \geq L_{\text{min}},
\]

\[
\text{Each VM should be placed on a PM : } \forall v_i \in V, \sum_{j=1}^{\lvert S \rvert} x_{ij} = 1,
\]

\[
\text{Number of active PMs : } |S| \geq y(t).
\]

It is easy to check that the above optimization problem is non-leaier and 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. Heuristic techniques have been used widely for solving optimization problems in an approximate way. The most common heuristic is First Fit Decreasing (FFD) \cite{10, 12, 43}. Recently genetic algorithms (GAs) \cite{44, 45} have shown success in tackling various classes of combinatorial problems, and grouping genetic algorithm has been shown to outperform the traditional heuristic techniques to solve VM assignment and packing problems \cite{46, 47}. Therefore we propose an improved grouping genetic algorithm (IGGA) to solve this optimization problem.
4. *RACE*: OPTIMIZATION APPROACHES FOR RELIABILITY-AWARE SERVER CONSOLIDATION

To efficiently explore the solution space, *RACE* applies an improved grouping genetic algorithm to approach the global optimum of reliability-aware server consolidation (named I$G^2$CA). In particular, as the feasibility of the initial solutions is critical to ensure the quality of final solution while improving the algorithm convergence rate, we also design a series of reliability-aware resource buffering and VMs-to-PMs re-mapping heuristics to generate sound initial solutions.

4.1. Applying an Improved Grouping Genetic Algorithm

Algorithm 1 shows our major procedures in I$G^2$CA. It begins with a set of initial solutions named *population*, which undergo crossover and mutation of VMs-to-PMs mapping configurations ($\chi$), and progressively converge to the optimal solution. We use the configuration $\chi$ to encode consolidation 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. 3.4) of the configuration $\chi$, as it represents the profit of the configuration. Then, the *selection probability* $P(\chi_i)$ of a solution $\chi_i$ is defined as the ratio of its individual fitness to the sum of fitness of whole population, given in Eq. (15), 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. Unfit solutions will be eliminated from the population. 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.

$$P(\chi_i) = \frac{F(\chi_i)}{\sum_{j=1}^{N_P} F(\chi_j)}, \quad F(\chi_i) = U(\chi_i). \quad (15)$$

Specifically, Algorithm 1 first initializes the system parameters and then generates initial consolidation solutions according to Algorithm 2. Secondly, Algorithm 1 selects two initial configurations $x$ and $y$ for crossover $\text{Crossover}(x, y)$, and the selection $\text{Select}(P)$ is based on the fitness probability according to Eq. (15). The crossover operation will generate $N_P \times R_c$ new configurations, where $N_P$ is the population size and $R_c$ is the crossover rate. Similarly, Algorithm 1 will select configurations for mutation and generate $N_P \times R_m$ new configurations. The fitness-based selection function $\text{ElitistSelection}(P, \text{NewConfiguration})$ is used to increase the concentration of good groups and avoid the loss of potential solutions in the successive generations. Unfit solutions will be eliminated from the population $P$. Finally, after $N_g$ times of iteration for crossover and mutation, the fittest solution will be selected as the final near-optimal configuration.

In sum, the complete design and major components of *RACE* is shown in Fig. 4. By periodically monitoring status of VMs and PMs like their resource demands and usages, *RACE* follows our utility...
Algorithm 1: Improved Grouping Genetic Algorithm for Server Consolidation (IG²CA).

I. Initial IG²CA

\[ N_P = \text{Population size of VMs-to-PMs mapping configurations}; \]
\[ N_g = \text{Number of iteration times to search solutions}; \]
\[ R_c = \text{Crossover rate of VMs-to-PMs mapping configurations}; \]
\[ R_m = \text{Mutation rate of VMs-to-PMs mapping configurations}; \]

// Generate initial solutions \(N_P\) using Algorithm 2 in Sec. 4.2;

\[ P = \text{InitialConsolidationSolutions}(N_P); \]

II. Iteration IG²CA(\(N_P, N_g, R_c, R_m, P\))

for \(i = 1\) to \(N_g\) do

// Select two initial configurations for crossover based on the fitness probability according to Eq. (15);

for \(j = 1\) to \(N_P \times R_c\) do

\((x, y) = \text{Select}(P);\)

\[ \text{NewConfiguration}[j] = \text{Crossover}(x, y); \]

// Select one configuration for mutation;

for \(k = 1\) to \(N_P \times R_m\) do

\[ z = \text{Select}(P); \]

\[ \text{NewConfiguration}[k + N_P \times R_c] = \text{Mutation}(z); \]

\[ P = \text{ElitistSelection}(P, \text{NewConfiguration}); \]

III. After iterating \(N_g\) times, the configuration \(\chi_{opt}\) with greatest selection probability is the final consolidation solution.
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. Also, we prefer to shut down the PMs with long running time. Accordingly, we define a metric called age-load-quotient (ALQ) in Eq. (17) to characterize the Load (defined in Eq. (5)) and Age status of a PM \( s_j \), where \( v = 3 \) (years) based on the empirical “bathtub curve”. \( RACE \) 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.

\[
ALQ_j = (\text{Age}_j - v)^2 / (\text{Load}_j \cdot T_j).
\] (17)

4.2.3. 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 (defined in Eq. (17)) in an ascending order to form an ALQ list. For each VM in the migration list, \( RACE \) attempts to 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. 3.2. It then migrates the VM to a PM that suffers from the minimized estimated temperature difference. In this way, \( RACE \) can eliminate hotspots, balance loads and maintain reliability of active PMs.

In sum, Algorithm 2 gives the complete pseudo-code of the above reliability-aware heuristics for generating good initial solutions as input to Algorithm 1. Specifically, when the average Load of the datacenter is below a minimum threshold \( L_{\text{min}} \), \( RACE \) will start to decide whether to perform consolidation or not. \( RACE \) first sorts all PMs by ALQ in an ascending order, and 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 wear-out age. Then, \( RACE \) will predict the temperature and resource buffer of each VM in the candidate PMs to be turned off, so as to find target PMs with the minimized estimated temperature difference for hosting each VM. Finally, \( RACE \) will calculate the overall utility of each configuration according to Eq. (1). If the utility \( U < 0 \), then rejecting the configuration and the datacenter configuration remains unchanged. Otherwise, the configuration is feasible and outputting it as initial solution for Algorithm 1.

We run Algorithm 2 \( N_P \) times, the above heuristics are changed slightly to generate different solutions each time. When the average Load of a datacenter is below \( L_{\text{min}} \) based on our utility models in Sec. 3, \( RACE \) will make decisions of whether and how to perform consolidation. Within such a strategy, our reliability-aware heuristics can help improve the convergence rate of Algorithm 1 towards the global optimal consolidation, by providing good initial solutions and avoiding potential trap to a local optimum.

5. \( RACE \): PERFORMANCE EVALUATION

In this section, we evaluate the effectiveness, overhead and scalability of our reliability-aware server consolidation strategy through a wide range of simulation experiments.

5.1. Experimental Setup

We simulate a datacenter based on an open-source discrete-event simulator CloudSim [25]. It can offer us a repeatable and controllable environment. Specifically, we simulate 100 PMs with \( P_{\text{idle}} = 185 \) W and \( P_{\text{max}} = 300 \) 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 well known 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. II.
Algorithm 2: Reliability-aware Heuristics for Generating Feasible Initial Solutions.

**Input:** Current configuration $\chi_{\text{cur}}$; multi-dimensional resource utilization and capacity of PMs $\theta_{jk}, C_{jk}$, where $k$ represents the resource types of CPU, memory, disk and network; current statistics of on-off cycles $\text{Count}[|S|]$, ages $\text{Age}[|S|]$, running time $T_i$ and $\text{MTTF}$ of PMs; $w$ previously measured resource demands.

**Output:** New configuration $\chi_{\text{new}}$, overall utility $U(\chi_{\text{new}})$; VM migration map $\text{MigMap}$ (which VMs assigned to which PMs).

$\chi_{\text{new}} = \chi_{\text{cur}}$.

Compute average $\text{Load}$ of a datacenter using Eq. (5);

if average $\text{Load} < L_{\min}$ then

Compute the $\text{ALQ}$ of PMs using Eq. (17);

Sort all PMs in $S$ by $\text{ALQ}$ in an ascending order;

for $s_j \in S$ do

if $\text{ALQ}(s_j) > \lambda_{\text{ALQ}}(\text{Threshold})$ then

// $\text{NodeOff}$ is a list of candidate PMs to be turned off;

$\text{NodeOff} \leftarrow s_j; S = S - \text{NodeOff}$;

end if

end for

Sort the VMs from the PMs in $\text{NodeOff}$ by their resource demands in a decreasing order, stored in a $\text{VMOff}$ list;

for $v_i \in \text{VMOff}$ do

for $s_t \in S$ do

Predict $s_t$'s temperature $T_{st}$ and its resource buffer $\text{Buffer}_{st}$ using Eq. (10) and Eq. (16);

if $(\text{Count}[s_t] < N_{\max})$ and $(T_{st} < T_{\max})$ and $((\text{Buffer}_{st} + \sum_{v_i \in s_t} R_i) < C_t)$ then

$\text{FeasiblePMs} \leftarrow s_t$;

end if

end for

if $\text{FeasibleServers} \neq \phi$ then

// Find target PM $s_o$ with the minimized estimated temperature difference for hosting $v_i$;

$s_o \leftarrow \text{FindTargetServer}(\text{FeasibleServers})$;

// Migrate $v_i$ to $s_o$, and update $\chi_{\text{new}}$;

$\text{MigMap}.\text{add}(v_i, s_o)$; $x_{oi}^{\text{new}} = 0; x_{io}^{\text{new}} = 1$;

else

break; // Stop re-mapping process;

end if

end for

Calculate the overall utility according to Eq. (1);

if $U(\chi_{\text{new}}) < 0$ then

Reject the configuration, the datacenter remains unchanged;

else

return $\chi_{\text{new}}, \text{MigMap}, U(\chi_{\text{new}})$ as input to Algorithm 1.

end if

---

Table II. Configuration of representative types of VM instances [48].

<table>
<thead>
<tr>
<th>Type</th>
<th>CPU (core)</th>
<th>Memory (GB)</th>
<th>Storage (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>1</td>
<td>1.7</td>
<td>160</td>
</tr>
<tr>
<td>Large</td>
<td>2</td>
<td>7.5</td>
<td>850</td>
</tr>
<tr>
<td>Extra-Large</td>
<td>4</td>
<td>15</td>
<td>1,690</td>
</tr>
</tbody>
</table>

A datacenter broker generates user requests in a Poisson process with an arrival rate $R_{\text{in}}$. We emulate light, normal, and intensive application workloads, denoted as $\text{App}_L$, $\text{App}_N$, and $\text{App}_I$, respectively, with different arrival rates $R_{\text{in}} = 5, R_{\text{in}} = 15$ and $R_{\text{in}} = 30$ requests per second. The request size is defined as the processing time of the request, following a bounded random
3.1 server consolidation can indeed incur increasing SLA violations. We then calculate the SLA violations of different consolidation strategies: pMapper [7] and PADD [11]. We also consider the base case without consolidation as a performance baseline. pMapper places VMs on the PMs taking into account the power and migration costs in addition to the performance benefit. PADD migrates VMs and packs them dynamically onto fewer PMs when possible with adaptive buffering scheme.

5.2 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 AppL, AppN, and AppI for 30 time intervals on the three representative VM instances in Table II. 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. 5, 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 RACE, the performance SLAs under RACE outperform 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. 6 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 the three types of consolidation strategies can save energy costs compared to the base case, especially for light and normal application workloads. While PADD places 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.

Copyright © 2013 John Wiley & Sons, Ltd.

and hence the former achieves higher cumulative profit than the latter does for most cases.

For each of these policies, we compare three different buffering schemes while fixing other parameters in Fig. 8: (1) No buffer. Each physical machine is allowed to be filled up to 100% of its capacity. (2) Fixed buffering. Each physical machine leaves 20% resource capacity as a resource buffer. (3) Adaptive buffering. The buffer of each physical machine is dynamic as defined in Eq. (16). For each of these policies, we emulate a datacenter that runs normal application workloads $\text{App}_N$ on large VM instances for 30 time intervals. The results show that the policies with buffer can significantly reduce the SLA violations but increase energy consumption as expected. Fixed buffering has lower SLA violations than adaptive buffering but has higher energy consumption and overall utility. The rationale is that fixed buffering has larger reserved resource and can reduce the resource deficit when the applications increase resource usage, but the reserved resource may not be fully used and incurs higher energy consumption.

Then, we proceed to take a closer look at the effects of different settings of respective utility weights $\lambda_{SLA}, \lambda_r$, and $\lambda_e$ in our multi-objective utility model in Sec. 3. Essentially, they represent the design preferences of a datacenter over different combinations of performance SLA, reliability and energy concerns. Specifically, we examine seven representative cases, including $RACE$ with $\lambda_{SLA}/\lambda_r/\lambda_e = 1/1/1$, $\lambda_{SLA}/\lambda_r/\lambda_e = 1/0.5/0.5$, $\lambda_{SLA}/\lambda_r/\lambda_e = 0.5/1/0.5$ and $\lambda_{SLA}/\lambda_r/\lambda_e = 0.5/0.5/1$, compared with $\lambda_{SLA}/\lambda_r/\lambda_e = 0/1/1$, $\lambda_{SLA}/\lambda_r/\lambda_e = 1/0/1$, $\lambda_{SLA}/\lambda_r/\lambda_e = 1/1/0$, 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 \textit{App}_N on small VM instances for 30 time intervals. Fig. 9 compares the cumulative overall utility over time under different settings.

We observe that \textit{RACE} achieves increasingly higher overall utility over time than the other settings. The rationale is that \textit{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 the 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. Further, although the cases $\lambda_{SLA}/\lambda_r/\lambda_e = 1/0.5/0.5$, $\lambda_{SLA}/\lambda_r/\lambda_e = 0.5/1/0.5$ and $\lambda_{SLA}/\lambda_r/\lambda_e = 0.5/0.5/1$ consider all the three objectives of performance, reliability and energy, but only the setting with $\lambda_{SLA}/\lambda_r/\lambda_e = 1/1/1$ has the highest cumulative utility. Therefore, we should treat performance, reliability and energy equally for maximized utility and avoid biased tradeoff among the three objectives.

5.4. Validation of Overhead and Scalability

Finally, we validate the overhead and scalability of \textit{RACE}. Specifically, by running the three types of application workloads \textit{App}_L, \textit{App}_N, and \textit{App}_I on small VM instances, we record the time interval between their start and end time as the execution time when using \textit{RACE}. This is compared with an optimal case by setting the number of iteration times $N_o$ of IG$^2$CA in Sec. 4 to infinity, which keeps searching the best solution until no greater utility can be brought forth. The overhead of using \textit{RACE} and optimal case is defined as the additional execution time of application workloads compared to that without using them, as shown in Fig. 10. Benefiting from our reliability-aware heuristics for generating good initial solutions in Sec. 4.2 and fine-tuned parameter settings in Sec. 5.1, \textit{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. 3.4, the solution space grows exponentially with the number of PMs and VMs. To improve the scalability of \textit{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 2, while the global controller applies IG$^2$CA to reconfigure the whole system less frequently (e.g., every dozens of minutes or hourly). The execution time of workloads (consolidation overhead) is shown in Fig. 11. We observe that \textit{RACE} takes less than 90s to consolidate workloads on 1,000 PMs. As \textit{RACE} can reduce the search space via reliability-aware heuristics and a distributed implementation, the execution time of using \textit{RACE} can be approximately maintained as a linear
6. LIMITS AND FUTURE WORK

Now, we discuss the main simplifications and limits of our approach.

**Server Heterogeneity:** In our model, different physical machines have different capacity of resource with respect to CPU processor, memory, disk and network. However, we assume that different physical machines have the same idle and peak power consumption, i.e., \( P_{idle} = 185 \text{ W} \) and \( P_{max} = 300 \text{ W} \). Actually, servers from different generations and vendors have very different power consumption.

**Comparison of the Optimal Solution and Competitive Algorithms:** We build a utility model to characterize the performance SLA, reliability impact, and power consumption of server consolidation. However, for the multi-objective optimization problem, we resort to a heuristic method—improved grouping genetic algorithm. We do not solve the problem use classical optimization techniques, and do not give a performance bound analysis for our algorithm. Future work consists of solving the problem using classical optimization techniques such as non-leaner programming and gives a competitive algorithm with performance bound with respect to the optimal solution, such as 3-competitive online algorithm \([23]\) and Lyapunov optimization techniques \([30]\).

**Real-world Experiments:** We evaluate our algorithm using simulation based on the Cloudsim simulator to simulate our cloud platform. However, the simulated workload is regular and has less stochastic demand than the real-world workload. Further, the interference and interactions among different component in the simulated system are ignored. The real-world experiment environment is more complex and may give certain surprising results. For example, the data transferred between two components in the Cloudsim is directly accessible, but in the real-world, the data may be blocked in the network. Future work consists of doing real-world experiments with more convinced results.

7. CONCLUSION

This paper presented RACE, a reliability-aware and energy-efficient 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 energy 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 traditional pMapper and PADD strategies that solely focused on the performance-energy tradeoff in an aggressive manner. Furthermore, we...
carried out simulations using the open-source CloudSim tool to show that, RACE is able to be deployed in large-scale datacenters with acceptable overhead.

ACKNOWLEDGEMENT

The research was support in part by a grant from The National Natural Science Foundation of China (NSFC) under grant No.61370232 and No.61133006, by a grant from the Research Fund of Young Scholars for the Doctoral Program of Higher Education, Ministry of Education, China, under grant No.20110142120079, by the CHUTIAN Scholar Project of Hubei Province.

REFERENCES


