International Journal of Grid and Distributed Computing
Vol.6, No.6 (2013), pp.63-82
http://dx.doi.org/10.14257/ijgdc.2013.6.6.06

Autonomic Power Aware Cloud Resources Orchestration Architecture for Web Applications

R. R. Darwish

Department of Mechanical Engineering, Helwan University, Cairo, Egypt

raniaa04@yahoo.com

Abstract

Endless resources provisioning illusion is the mainstay for cloud computing paradigm. However, the unpredictable volatility nature involving web applications workload demand would highly hinder cloud computing platforms performance, furthermore, expose cloud resources for possible devastation. Accordingly, this work proposes autonomic power aware SLA-oriented cloud resources orchestration two-tier architecture. Despite complexity and uncertainties of the workload fluctuations, the proposed architecture geared for leveraging cloud system resources utilization, ensuring explicit guarantees on web applications' responsiveness obligations, meanwhile achieving power consumption minimization objectives. The proposed architecture consolidates heuristic methodologies along with control theory approaches in a resource orchestration hierarchical structure. Firstly, an autonomic global controller is presented. The proposed global controller exploits heuristic methodology for mapping virtual machines (VMs) to the appreciate cloud resources in accordance to heuristic multidimensional objectives based placement strategy. Secondly, a proactive fuzzy-logic based local controller is proposed. The proposed local controller aimed at in confronting workloads' sustainable fluctuations via proactive amendment for the placement and provisioning schedules. Furthermore, the proposed local controller oriented towards maintaining active power management policy especially during transient peak of usage, thereby mitigating overall costs, and extending resources capacity and performance capabilities. Simulation results and comparisons demonstrate that the proposed architecture significantly surpasses previous approaches in terms of total energy consumption, furthermore maintaining web applications SLAs objectives despite dynamic workload scenarios.

Keywords: *Bees Algorithm, Cloud Computing, Fuzzy Logic Controller, Resource Orchestration, Virtualization, Web Applications*

1. Introduction

Cloud computing has become a significant technology trend that inherited the legacy technology while adding new ideas. Many experts argue that cloud computing addresses the next evolutionary step of distributed computing that will reshape information technology (IT) processes and the IT marketplace [1, 2]. The vision for this computing model powers data centers consolidation by architecting them as a pool of virtualized, dynamically-scalable services (computing power, storage, platforms) to be delivered on demand to external customers over the Internet according to Service Level Agreements (SLAs) [3, 4].

Cloud computing paradigm imposes new challenges. The most prominent being, is the illusion of infinite computing and storage resources. Meanwhile the ability to provide scalable, secure, and autonomic services to support ubiquitous, elastic (dynamically growing and shrinking) demands, while keeping the users isolated from the underlying infrastructure [3, 5, 6]. In such complex environment, virtualization emerges as a core technology. Employing virtualization increases the utilization potential of a physical server to multiple users, thus allow better resource utilization. Virtualization also enables customization of the platform to suit diverse needs of the end users, while providing the adequate isolation. Furthermore, adopting virtualization assists at increasing application availability, and providing required configurability, manageability, and better security [1, 3, 5].

Nevertheless, employing cloud computing for the virtualization concept raises further challenges. As a matter of fact a typical workload in such cloud infrastructure would comprise a dynamic mix of heterogeneous applications. Those applications may contain long running computationally intensive jobs, data and IO-intensive analytics tasks, moreover, bursty and response-time sensitive requests. Therefore, intelligent allocation of physical resources for managing competing resource demands of the users is inevitable. A typical scenario is a statically-configured virtual resource. However, the increasing difference between these resource allocations and application requirements especially in case of web applications where workloads fluctuate considerably will lead to either over-provisioned or over-loaded VMs [6, 7]. Furthermore, it rather to be mentioned that power has an immense potential at theses large scale data centers. Costs for power and cooling capabilities are becoming increasingly large. Thus, lowering power consumption could improve cost effectiveness. Moreover, active management for power consumption could assist at dealing with possible limitations that may take place at cooling and power delivery capabilities resulting from data centers' attempts for meeting the ever increasing performance and scalability demands of enterprise applications [8, 9]. According to the foregoing, drawn to autonomic cloud resources management and orchestration frameworks that capable at trading between better resources utilization, maintaining the desired application's SLA obligations, meanwhile, supporting active power budgeting constrains, is an inevitable challenge.

This paper tackles resource orchestration problem for web-based cloud services. An autonomic power aware SLA-oriented two tier hierarchical resource orchestration architecture is proposed. While coping with complexity and uncertainties of web services' workload fluctuations, the overwhelming goal of this work is also developing an orchestration architecture capable at maintaining targeted web-applications responsiveness obligations, elevating resources utilization, meanwhile, budgeting overall power consumption. The proposed work integrates heuristic autonomic VM placement, and planning tools along with power-aware proactive resources provisioning strategies. Firstly, a heuristic global controller manages VMs mapping decisions among various nodes satisfying multidimensional placement objectives. Adopted Placement objectives comprise confirming workload resources requirements (specifically CPU, and memory constrains). Meanwhile, satisfying security, reliability load balancing constrains. Secondly, a proactive local controller is incorporated at each physical server. Local controller is aiming at maintaining resident applications responsiveness objectives, meanwhile, attaining power consumption policies despite internet workload uncertainties nature, or any other exceptional situations that may occur. Proposed local controller encompasses fuzzy logic controller, augmented with feed forward predicative performance scheme, along with on line monitoring module. According to the VMs' predicted performance information, and monitored resources usage statistics, the proposed fuzzy controller decides the appreciated action for facing either the time varying demand workload fluctuations, possible hotspot occurrence, server failure, or idle situation that may occur. Admitted possible actions for the local controller endures either alerting the recourses being provisioned to the resident VMs through VMs resizing, exploiting Dynamic

Voltage/Frequency Scaling (DVFS) server capabilities, initiating candidates for VMs live migrations decisions, or turning off the managed server.

The continuation of this paper will be made up as follows: Section 2 outlines the related work to the theme of the paper. Section 3 shows the preliminaries for the adopted model. Section 4 presents the proposed architecture. Section 5 demonstrates the performance evaluation of the proposed architecture. Finally, Section 6 concludes the paper.

2. Background

Recently, cloud computing has received much attention at the research community. Quiroz *et al.*, [7] explored autonomic approaches for optimizing VM provisioning in grid and cloud environments. The authors proposed an online decentralized clustering approach for VM provisioning capable at characterizing dynamic classes of resource requirements and can be used for proactive VM provisioning. Also, the authors demonstrated a model based approach for estimating the application service time given its provisioning in order to deal with inaccurate resource requirements provided by application job requests.

Bi *et al.*, [10] proposed an autonomic virtual machine provisioning architecture for multi-tier application in cloud data center. The proposed work employed a flexible hybrid queuing model to minimize the total number of virtual machines at each tier of virtualized applications.

Tang *et al.*, [11] tackled the problem of Software as a Service (SaaS) components' placement in the cloud. The authors introduced a Genetic-based placement algorithm for the composite SaaS. The proposed algorithm concerned with the placement of both the component's data as well as software components of the SaaS. The authors showed that the proposed algorithm ensured optimizing SaaS performance, meanwhile guaranteed well placement for the SaaS software components and the components' data as well. However, the algorithm is centralized, and the problem has not been explored in the context of the optimization of energy consumption.

Zhu *et al.*, [12] provided dynamic resource provisioning algorithm for adaptive applications in cloud environment. The authors developed a feedback control approach aimed at maximizing application QoS, taking into consideration both time constraint and resource budget limit (CPU, and memory). However, similarly to [11] the proposed approach does not consider energy management.

Carrera *et al.*, [13] provided adaptive resource allocation and dynamic load balancing prototype for Web applications in the cloud. In order to satisfy the SLA terms that enforces specific response time requirements, the proposed approach monitored the response time of the computer resources assigned to a Web application, and then dynamically allocated the resources required by the application. However, the proposed prototype is only able to scale up the applications, taking into consideration CPU management only while neglecting other system resources. Also, the proposed prototype didn't guarantee satisfaction of the SLA terms as it would be better employing response time violation predicting module rather than waiting until the requirement is violated. Meanwhile, optimizing energy consumption was not considered.

Chieu *et al.*, [14] presented a dynamic scaling scenario for cloud web applications. The authors considered using scaling indicator in order to track the number of current users and the number of current sessions at each web application. Depending on the updated statistics, action to scale up or down would be initiated. However, the work uses simple triggers and thresholds to determine instance number but does not consider VM type information and budget constraints as well. Meanwhile, optimizing energy consumption is not addressed in this work.

Srikantaiah *et al.*, [15] presented an energy efficient heuristic approach to perform workload consolidation of multi-tiered web applications in cloud environment. The authors discussed consolidation as a modified multi-dimensional bin-packing problem of allocating and migrating workloads to achieve energy optimal operation. However, this work assumed homogenous environment which is not the case in the cloud. Also, this work focused only on CPU and disk resources and neglected many other issues that affect consolidation, including application affinity, server and workload behavior, security restrictions that required co-location of certain application components. Furthermore, this work did not provide an explicit mechanism for amending resources provisioning to cope with the frequent work load fluctuations which considered a salient feature at web applications.

Buyya *et al.*, [16, 17] introduced architectural principles for energy-efficient management for cloud environment. First, the authors proposed an energy efficient VMs mapping policy using a modification of the Best Fit Decreasing (MBFD) algorithm. In addition, the authors proposed dynamic VM consolidation algorithms. The basic idea of the proposed consolidation algorithms depends on setting either a fixed Single Threshold (ST) [17], or setting fixed CPU upper and lower utilization thresholds for hosts [16]. Such that hosts with CPU utilization that falls below the lower CPU threshold is turned to a sleep mode, and all VMs that had been hosted by that host might be migrated. While hosts with CPU utilizations exceed the upper threshold, some of their VMs had been selected for migration to reduce the overall utilization. Finally, the author presented three heuristics for selecting the VM/s to be migrated: Minimization of Migrations (MM), Highest Potential Growth (HPG), and Random Choice (RC). However, the proposed architecture only issued CPU utilization, neglecting any further placement constrains for example other physical resources constrains, VMs types, and security requirements. Also, the proposed architecture didn’t address optimizing VMs placement in the context of minimizing network infrastructure energy consumption, and network transfer overheads. Furthermore, the proposed consolidation approach is based on setting fixed upper and lower CPU utilization thresholds which does not suit cloud environments with mixed workloads that exhibit non-stationary resource usage patterns.

Apart from all these efforts, our work is noteworthy at several aspects: Firstly, the proposed approach addresses various VMs placement constrains that cover a much wider scope than what previously proposed. The considered placement constrains are ranging from applications affinity, physical resources capacity, to security requirements. Meanwhile, the communication patterns between various VMs have been also considered, thereby substantial minimization at network infrastructure energy consumption, and network transfer overheads could be reached. Secondly, our approach incorporates mechanisms for maintaining even load distribution across servers and therefore enhances robustness and reliability. Thirdly, the proposed approach shows an explicit consideration for work load demands volatility problem, as fuzzy logic feed forward approach is demonstrated for proactively amending the provisioning strategies in order to remedy exceptional workload fluctuations that frequently occur at runtime. Finally, our work provides active power budgeting management to curtail overall power consumption. Moreover, guarantees application-level SLAs obligations, specifically the response time of the web requests.

3. Preliminaries

3.1. System Model

Our hosting cloud model represents the underlying infrastructure as a large-scale data center hosting various web applications. The assumed data center comprises ***N*** heterogeneous physical servers, where physical resources are shared among hosted applications in a virtualization environment. Each physical server holds virtual containers that will host users' requests. Similar systems have been investigated in for example [16-19]. Users submit requests for provisioning ***M*** heterogeneous VMs. Each with a defined priority, resource requirements expressed as Millions Instructions Per Second (MIPS), and specific amount of RAM, along with never/always together VMs list. In order to determine workload requirements, this work assumed that a priori profiling stage is employed, where VMs are first profiled on a staging server to determine the amount of physical resources needed to achieve the negotiated SLA-level. Also, during this stage each submitted VM is assigned a predefined priority that is comes from how critical is this VM. For example, critical business applications take higher priority than the ordinary web applications. Each request can be treated independently of other requests. Each request has an SLA defining the QoS that the application should guarantee from the cloud system. In this work the SLAs contain the average response time for each request, meaning that an application should have a specific share of the CPU capacity thus its clients experience an acceptable response time from the cloud system. SLA violation occurs when a VM cannot get the requested amount of resources.

3.2. Energy Model

Power consumption by computing nodes in data centers is mostly determined by computation processing, memory, network interfaces, and cooling systems. However computation power contributes significantly at the overall datacenter's power consumption [16, 20]. The power consumption for a CMOS circuits composed of dynamic power consumption $P_{dynamic}$, and static power consumption P_{static} [21]:

$$P = P_{dynamic} + P_{static} \qquad (1)$$

According to [22], the dynamic power consumption $P_{dynamic}$ can be represented as [22]:

$$P_{dynamic} = C_{eff} . V_{dd}^2 . f \qquad (2)$$

Where C_{eff} is the effective switching capacitance, V_{dd} is the supply voltage, and f is the processor clock frequency. Since the processor clock frequency f is usually in proportion to the supply voltage and could be expressed as [22]:

$$f = k.(V_{dd} - V_{th})^2 / V_{dd} \qquad (3)$$

Thus, dynamic power of a processor $P_{dynamic}$ could also be expressed as [20]:

$$P_{dynamic} = C_{eff} . f^3 \qquad (4)$$

Static power consumption is primarily occurred due to leakage currents (I_{leak}), and it could be expressed as [23]:

$$P_{static} = I_{leak} . V_{dd} \qquad (5)$$

Finally, the total energy consumption by a physical node can be defined as the integral of the power consumption function over a period of time and can be expressed as follows [16]:

$$Energy = \int_{t_o}^{t_1} P(t) \cdot d(t) \qquad (6)$$

It worth to be noted that throughout this work, a simplified power and energy models expressed in eq. (7), that previously used in [24, 25], have been adopted.

$$\begin{aligned} P &= f^3 \\ E &= f^2 \end{aligned} \qquad (7)$$

4. The Proposed Architecture

This section provides an overall description for the proposed architecture. As previously mentioned, the major motivation behind this work is proposing a resource management and capacity planning architecture capable at addressing the unique web applications features. However coping with salient features of web workload demands would entail new aspects for cloud developers: Firstly, those workload demands are inherently non-uniform that drastically volatile over short and long periods of time. Secondly, being exposed to unexpected request patterns makes predicting the workloads behaviors nontrivial task. Finally, availability and reliability are crucial factors for web applications' users. Despite of the number of concurrent accessing is, the fastest response time with no absolutely downtime is a pivotal requirement for web applications' users. These entire factors make the problem of resource orchestration is non-trivial task. Unless using dynamic and autonomic resources orchestration approaches that capable at accommodating such dynamic and unpredictable workload fluctuations, system resources would be devastated and contracted SLAs terms would be violated. Consequently this work proposes autonomic power aware SLA-oriented resource orchestration two-tier architecture. The proposed architecture specifically tackles the challenges in offering explicit guarantees on both power consumption and applications' obligations. Through leveraging efficient algorithms, the proposed architecture is capable at:

- Mapping the requested VMs to the appreciate cloud resources in accordance to multidimensional mapping objectives. Those objectives confirm load balancing policy, meanwhile align with cloud resources capacity and applications requirements.
- Self managing sustainable fluctuations in applications' workloads via proactively amending VMs placements and provisioning schedules in a manner that satisfy performance obligations.
- Maintaining active power management policy especially during transient peak of usage thereby mitigating overall costs, and extending resources capacity and performance capabilities.

Figure 1 depicts an architectural overview of the proposed system components which will be discussed with explicitly later on the next subsections. Firstly, the figure shows *resource broker*, that analyzes the submitted requests, also acts as a profiling stage. Then, the figure shows the proposed tiered management architecture that comprises *local* and *global controllers*. Finally, the figure shows the managed infrastructure, where each physical server runs multiple virtual machines each hosts a specific request.

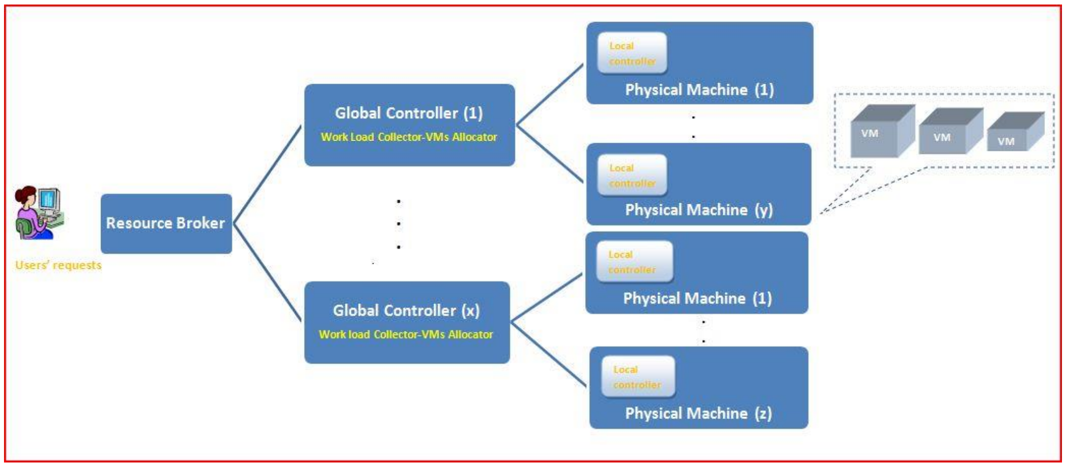


Figure 1. The proposed cloud orchestration architecture

4.1. Global controller

The main objective for proposed global controller is to determine the appreciate placement decisions for both the newly requested VMs submitted by customers to the cloud servers, and those chosen to migrate across attached physical servers. The proposed methodology phrases this issue as a constrained optimization problem that defines a cost function favors attaining balance between predefined placement objectives. Adopted methodology's placement decisions exploit both server capacity and utilization statistical information that is relayed from local mangers, and the predetermined resources requirements formulated during the profiling stage. Besides finding a viable configuration that is to satisfy VMs' physical requirements for achieving a certain level of SLA without exceeding physical servers' capacities, adopted strategy imposes other placement objectives for the submitted VMs configurations:

Managing communicating Virtual machines: As previously mentioned, this work is interested in cloud internet applications which are typically communication oriented applications adhere to highly modular software architectures with multiple communicating tiers. Thereby, placing the communicating VMs in hosts at separate enclosures will result in significant energy consumption, and further overhead will be inured. Thus, our approach tries to optimize the allocation of communicating VMs through taking into consideration the VMs communication behavior during managing the placement decisions process, leading to placement configuration plan that mitigate both network transfer overhead, and communicating infrastructure energy consumption.

Virtual machines affinity rules: Affinity rules regulate VMs placement in such a way that is to permit satisfying certain security and performance constrains, and to deliver a higher level of availability. According to those rules a group of VMs could be set to be always placed on the same host. On the contrary, other group of VMs according to those rules may be forbidden to be placed on the same host. Thus our approach aiming at finding a viable VMs placement configuration governed by application affinity rules those that is addressed and expressed at ***never/always*** together lists.

Virtual Machines priorities: the proposed placement approach is also aiming towards maintaining an even distribution for critical VMs over the cloud servers.

Proposed global controller utilizes the Bees algorithm [26] as an optimization tool, where appropriate fitness function is developed to incorporate the above-described optimality criterions. Bees Algorithm is a population-based optimization algorithm that mimics the food foraging behavior of swarms of honey bees. In its basic version, the algorithm performs a kind of neighborhood search combined with a random search and can be used for both combinatorial optimization and functional optimization [26]. Figure 2 shows the applied pseudo code for the algorithm. The algorithm requires a number of parameters to be set, namely: number of scout bees (n), number of sites selected out of n visited sites (s_e), number of best sites out of s_e selected sites (e), number of bees recruited for best e sites (nep), number of bees recruited for the other selected (m-e), and selected sites (nsp).

```
1.      Generate initial population of n random solutions;
2.      Assign fitness function to each individual;
3.      Store the fittest individual in the initial population Best;
4.      While (stopping criteria is not meet);
    // Forming a new population
5.      Select elite bees and elite sites for neighborhood search;
6.      Select other sites for neighborhood search;
7.      Recruit bees around selected sites and evaluate their fitness;
8.      Select the fittest bee from each site;
9.      Assign the remaining bees to search randomly;
10.     Assign fitness value to  each individual;
11.     Store the fittest individual $B_i$ in the $i^{th}$ iteration;
12.     If $B_i$ < Best;
          Begin
13.     Best=$B_i$;
        End;
14.       End while;
15.     Return Best;
```

Figure 2. Bees Algorithm [26]

4.1.1. Methodology of Bees Algorithm: This sub section illustrates the methodology and the formulation for Bees algorithm for obtaining the optimal VMs configuration. Proposed methodology comprises problem representation, and formulation of the fitness function that is to be given to each individual solution. Considering a hosting environment of **M** VMs, and **N** servers. Each bee in the population specifies a possible configuration for the VMs placement plan. Such configuration is encoded as a vector comprising d points, each of which corresponds to a certain VM. This vector can be interpreted as the order in which the VMs could be distributed along available cloud servers according to this possible placement plan (*i.e.*, each point in this vector represents the cloud server where the VM would be placed through this placement plan). For the case under investigation, the fitness function used to measure the quality of a specific configuration should reflect the above discussed placement objectives which to be optimized by the Bees algorithm. Therefore, the fitness function **Obj** could be described as "8". Table 1 lists all possible notations used through the rest of this paper.

$$Obj = \min\left(\left(\sum_{i=1}^{M} load_i^{cpu}.\ x_i^{cpu}\cdot T + load_i^{mem}\ .\ x_i^{mem}\cdot T + p_i\cdot Q\right) + Attrib_{affinity}\cdot H + Attrib_{comm}\cdot K\right)$$

$$Where: \qquad (8)$$

$$x_i^{cpu} = \begin{cases} 1 & if\ load_i^{cpu} > capcity_i^{cpu} \\ 0 & if\ load_i^{cpu} < capcity_i^{cpu} \end{cases}$$

$$x_i^{mem} = \begin{cases} 1 & if\ load_i^{mem} > capcity_i^{mem} \\ 0 & if\ load_i^{mem} < capcity_i^{mem} \end{cases}$$

Table 1. Possible notations

N	Number of physical servers
M	Number of VM requests
$capacity_i^{cpu}$	CPU capacity of the i^{th} server
$capacity_i^{mem}$	Memory capacity of the i^{th} server
$load_i^{cpu}$	Total allocated CPU load at the i^{th} server
$load_i^{mem}$	Total allocated memory load at the i^{th} server
P_i	Number of critical VMs assigned to server i that exceeds a predefined upper limit of critical VMs per server
$Attrib_{affinity}$	Overall number of VMs that did not obey affinity rules
$Attrib_{comm}$	Overall number of VMs that did not obey communication rules

It can be clearly seen from “8” that, the fitness function *obj* rewards configurations that satisfy adopted placement constrains, while infeasible configurations those that violated the placement objectives would be imposed a high penalties (T, Q, H, and K). Thus, infeasible configurations will be always having a lower fitness than other feasible configurations and hence will be discarded. So minimizing this function will provide a VMs placement configuration that best obey application affinity rules, guarantee even distribution for critical VMs, and decrease communication overhead, and communication infrastructure energy consumption through perceiving VMs communication behaviors. The value of T, Q, H, and K is determined heuristically, where increasing T, Q, H, and K decreases the possibility of accepting placement configurations that violate any of the predefined placement objectives.

4.2. Local controller

Bursty internet workload nature necessitates dynamic adaptation for the resources that is to be provisioned during runtime in order to ensure applications Qos level. Therefore, according to the proposed architecture each physical server is controlled by a local manager that is to attain the allocated applications’ objectives meanwhile minimize total power consumption under work load fluctuations or any other disturbance. Local manager attains its objectives through relying on control theory methodologies. The Rationale behind exploiting control methodologies initiated from the analytical assurance of control accuracy and system stability for control theory methodologies, also the ability of control theory methodologies to quantitatively analyze the control performance even when the system model changes significantly due to various system uncertainties [27]. Thereby, previously mentioned features make control theory methodologies represent a good candidate for fulfilling the local manager objectives although the volatility of application workload intensity.

This work proposes exploiting fuzzy logic based feed forward controller for amending resource provisioning and VM allocating regime in such a way that proactively react to observed quality degradation or any exceptional situation. Feed forward approaches usually combined with queuing theory. However, according to [28, 29] queuing model based systems have little applicability in highly transient workload conditions, as prediction through queuing theory relies only on long-term averages, thus is insensitive to sudden load changes. Therefore, our approach refuge towards utilizing a feed forward predictive performance scheme [30]. Feed forward predictive performance scheme directly relates instantaneous measurements of arrival times and queue length to the average delay over a finite prediction horizon, hence allowing finer grained prediction and achieving tighter control over traditional queuing theoretic prediction approach [30]. Along with feed forward predictive approach, the proposed controller scheme is also augmented with online monitoring mechanism for tracking processor, and memory usage for each hosted VM, meanwhile overall server resources' load situation is also tracked through aggregating the usage of the resident VMs. The proposed fuzzy controller feed with both predicted performance and monitored statistics, is then proactively initiates the appreciated action to remedy possible SLA violations, hotspot, server failure, or idle situations that may occur. Set of possible actions utilized at responding to exceptional situation: exploiting DVFS decisions, VMs resizing, live migration for one or more VMs, or turning off the managed server in case of low utilization. It worth to be mentioned that, utilizing a cloud server that support some actuation mechanism capable at managing DVFS decisions, VMs migration and resizing decisions is assumed. Figure 3 depicts the main modules comprising the proposed local controller which will be explained in more details in the subsequent sections.

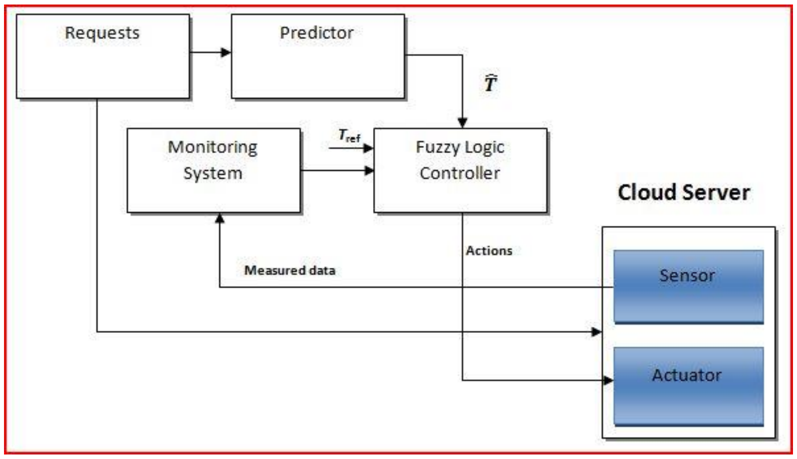


Figure 3. Proposed local controller architecture

4.2.1. Prediction model: Proposed controller module utilizes the performance predictor approach presented in [30], where instantaneous measurements of queuing delays and estimated processing times are used to predict and influence the average delay of future requests. As previously mentioned, this work assumes that several web applications reside on the same physical server, each assigned to a dedicated VM. Also, all incoming web application requests for each VM are forwarded to a VM's infinite queue, therefore processed in a First-In-First-Out fashion. Figure 4 depicts the main idea of the utilized performance predictor approach. It worth to be mentioned that, the total delay for a web request comprises a queuing delay, that is the request spends in the VM's queue, and a processing delay.

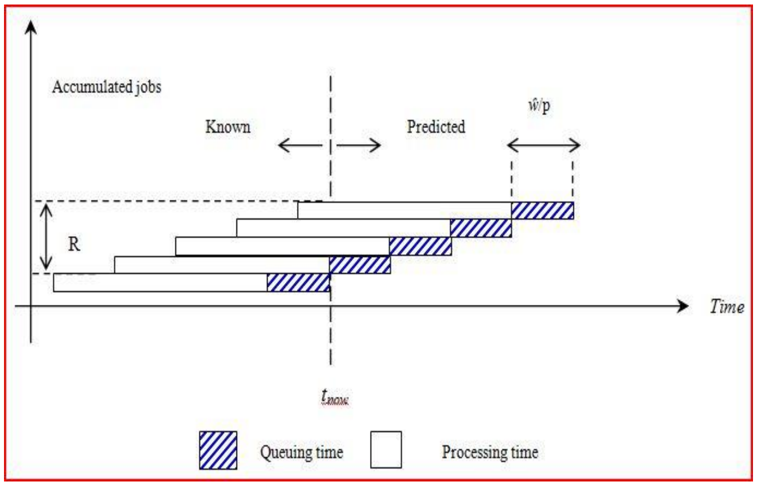


Figure 4. Accumulated jobs in a single server queue [19]

Considering the situation t_{now}, where most recent request departs the server leaving R of requests queued up waiting to be processed. Assuming that all web application's requests have an average nominal processing time $\hat{w}$, and a reserved CPU share p_r. So, time to process a request can be modeled as being inversely proportional to the reserved share of the CPU p_r [19]. Thus, By assuming that all requests of the same web application will have the same processing time $\hat{w}$, P_r still constant, prediction of the average response time $\hat{T}$ of a web request could be expressed as follows [19]:

$$\hat{T} = {}^{1}\!/_{N}\sum_{i=1}^{N}(t_{now} - a_i) + \hat{w}(N+1)/2p_r \tag{9}$$

Where a_i is the arrival time of request i. The first term on the right hand side of the equality sign represents the known area of the figure that can be calculated from measurements, and the second term represents the prediction.

4.2.2 Monitoring system: The monitoring system tracks the processor and memory usage for each VM. It also tracks the total resource usage on each physical server by aggregating the usage of the resident VMs. The monitoring engine tracks the usage of each resource over a measurement interval **I**, and reports these statistics to the control plane at the end of each interval.

4.2.3 Monitoring system: The proposed local control relies on fuzzy-logic feed forward control theory. In general, fuzzy controller widely utilized in control problems where it is difficult to construct precise mathematical models [30, 31]. For internet applications, these difficulties arose from inherent uncertainty, and workload demand volatility over short and long, furthermore modeling performance metrics along dynamic resource allocation exposed by virtualization technology is non trivial as virtualization allows available capacities to be changed overtime [18]. Thereby fuzzy controller represents an appreciated solution for the case under investigation where dealing with internet applications is the main concern. The main objective for the proposed local control module is maintaining the hosted applications' SLAs despite workload fluctuations. That is to keep the average response time below a reference value T_{ref}, meanwhile conforming power budgeting obligations. It has been assumed throughout this work that, the response time of a web application is independent from any other web applications. Also, arrival rate and number of requests for each internet application are available for measuring. The proposed controller module accepts two kinds of inputs, firstly, the predicted application response time $\hat{T}$ deduced from the prediction module. Moreover, monitored information delivered from the monitoring module that comprises entitlement CPU load, and overall power utilization level.

The proposed fuzzy controller's overwhelm objective is to deduce the proper action in order to maintain an actual average response time T_{actual} for the hosted applications close to the reference value T_{ref} despite applications work load dynamic fluctuations. The operation of the local controller is invoked periodically, thus its operation is partitioned into rounds. The following steps performed at the beginning of each round:

- The performance predictor module begins to compute the estimated response rate according to the measured request rate of each application.
- The monitoring module attached with each server measures the actual CPU entitlement load, also the overall utilization power level.
- Inputs for the modeling system comprising predicted response time $\hat{T}$, measured CPU load being provisioned, and power utilization level are (fuzzified) into the equivalent linguistic input variables using the membership functions of the linguistic variables.
- Fuzzy rules are then evaluated using the fuzzified inputs values.
- The centroid approach is then applied during the defuzzification phase on the derived decision function, which comprises crisp values representing the possibility of each action to be taken by the fuzzy controller in order to remedy the workload dynamic volatilities or exceptional situations.
- Finally, the local controller calculates the possibility of each action then selects the appreciate action either VM resizing, migrating one or more VM, exploiting DVFS decisions, or turning off the server in case of low utilization.

The proposed fuzzy logic feed forward controller defines a set of **30** fuzzy control rules. The general form of the applied control fuzzy rule that has a number of independent control variables can be expressed as:

Rule L: IF $\hat{T}$(I) is **a**, CPUload(I) is **b**, and Powerlevel(I) is **c** Then the action ***E*** Is applicable.

Where, Rule L is the L th control rule (L=1,2,….,l), **l** is the total number of control rules. $\hat{T}$ ***(I)*** is the predicted performance calculated at the beginning of the current controlling period **I**. Also, ***CPUload(I)***, and ***Powerlevel(I)*** respectively represent monitored total CPU cycles being entitlement, and overall power level in the current control period I. Finally **a, b,c** are fuzzy linguistic variables, **E** is the selected control action. The following represents sample fuzzy control rules.

Rule 1: If $\hat{T}$(I) is **Low**, CPUload(I) is **Low**, and Powerlevel(I) is **Low**, Then ***Expanding*** Is applicable.

Rule 2: If $\hat{T}$(I) is **Low**, CPUload(I) is **High**, and Powerlevel(I) is **High**, Then ***Migration*** Is applicable.

Rule 3: If $\hat{T}$ (I) is **High**, CPUload(I) is **High**, and (Powerlevel(I) is **High** Or Powerlevel(I) is **Medium**), Then ***Scaledown*** Is applicable.

The first sample rule states that it is desirable for handling application workload increase to expand the CPU share assigned to the VMs hosting resident applications, especially that there is enough free idle resources available at the physical server. The second rule states that it is reasonable to move one or more VM to another server as the hosting server is very loaded. Thus, chosen VM for migration is delivered to local control for selecting suitable server host. Finally, the third rule states that it is applicable to scale down the supply voltage one level as the request rate decreases in order to save total power consumed.

5. Simulation

This section presents the results of the conducted experiments analyzing the proposed architecture performance aspects. Conducted experiments have been oriented towards investigating the proposed architecture energy efficiency, expressed as the total energy consumption. Meanwhile, SLA's guarantees specifically response rate obligations for the proposed architecture. For that purpose, a complete environment comprises 100 heterogeneous server nodes is simulated. For the comparison sake, the same assumptions concerning servers specifications, virtual machines requirements addressed at [16, 17] are adopted. According to this model, each node is modeled to have a single CPU core with the performance equivalent to 1000, 2000, 3000 MIPS, 8 GB of RAM and 1TB of storage. Each virtual machine requires one CPU core with 250, 500, 750 or 1000 MIPS, 128 MB of RAM and 1 GB for storage. Also, in the adopted model it has been assumed that, the time needed to perform a live migration of a VM is calculated as the size of its memory divided by the available network bandwidth. This work also assumed that, each physical node in the simulated environment supports DVFS capabilities. Where each processor *p* can operate at different supply voltage levels within a defined range from $V_p^{\min}$ to $V_p^{\max}$ discretely. The associated processor speed with each supply voltage V_p^i is denoted as Q_i^P in terms of MIPS. The normalized speed of each voltage V_p^i is defined as S_p^i, which is determined by $\frac{Q_i}{Q_m}$. Finally, table 2 summarizes the main parameters of the exploited Bees Algorithm.

Table 2. Parameters of the Bees Algorithm

Bees Algorithm parameter	Symbol	Value
Population size	n	50
Number of selected sites	s_e	30
Number of elite sites	e	3
Number of bees recruited for elite sites	nep	30
Number of bees recruited for the other (m-e) selected sites	nsp	10
Number of iterations	itr	500

The proposed architecture's performance is investigated under both steady state and transient load behaviors. Firstly, Figure 5 depicts the average response rate for the proposed architecture along with Queuing-Theoretic Feed Forward based predictor (QFF) [32] at different steady state arrival rates λ. It can be seen that the proposed architecture, and the (QFF) both show good performance rate, that capable at maintaining a response rate close to the reference response rate under low and high medium rates. However, the proposed architecture shows a better performance at the high arrival rates than that of the QFF.

Increment

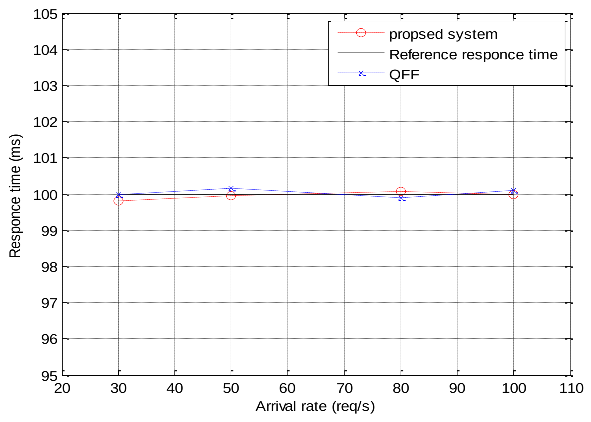


Figure 5. Response time of the proposesd architecture vesus QFF at different arrival rates

Secondly, two experiments have been conducted in order to validate the robustness of the proposed architecture under transient load behavior. Figure 6 illustrates the response rate of the proposed architecture in such a case where the arrival rate was initially set at 50 req./sec then at round number ten it has been doubled to be 100 req./sec. Also, Figure 7 depicts the response rate of the proposed architecture in such a case where the arrival rate was initially set at 100 req./sec then at round number ten it has been decreased to be 100 req./sec. Figure 6 and Figure 7 shows that, the proposed controller exhibit better performance than QFF at both cases sudden increasing and decreasing the arrival rate. The results show that, the proposed controller exhibit immediate response to the changes with a moderate increase at the response time, however, QFF did not react immediately with over reacting at the response time.

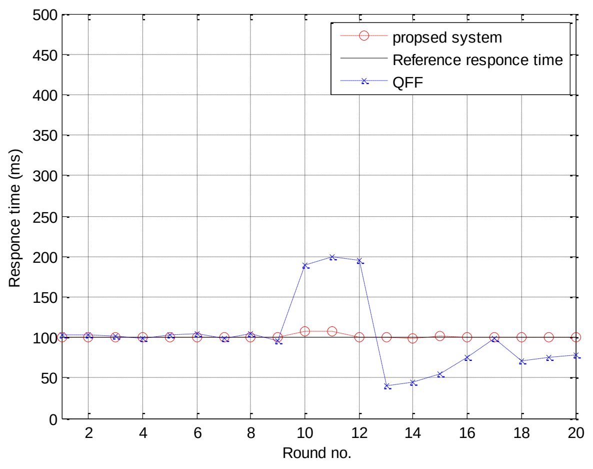


Figure 6. Respone time of the proposed architecture versus QFF with increacsing arrival rate

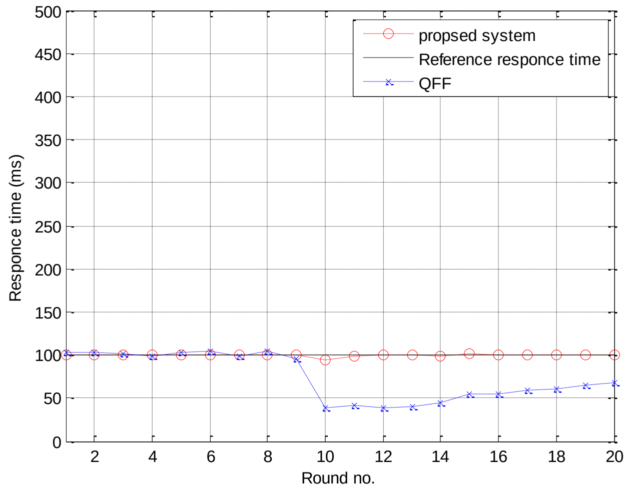


Figure 7. Respone time of the proposed architecture versus QFF with decreasing arrival rate

Figure 8 compares power consumption yielded at employing the proposed architecture along with those resulted when four bin backing heuristics algorithms [33] have been applied: First Fit Decreasing sorted by CPU requirements (FFD-CPU) and memory requirements, (FFD-MEM), and Best Fit Decreasing (BFD) algorithms also sorted by CPU requirements (BFD-CPU) and memory requirements (BFD-MEM). It can be seen that the proposed strategy capable at effectively minimizing total energy consumption among cloud servers in comparison to the considered static bin backing heuristics algorithms.

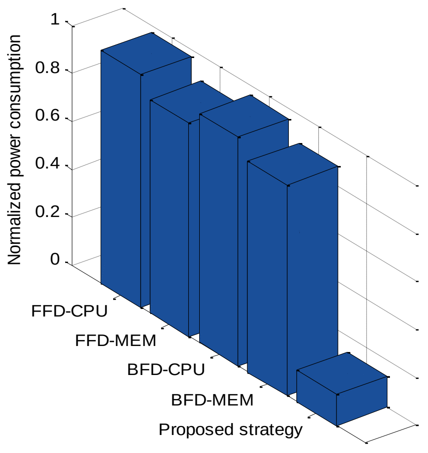
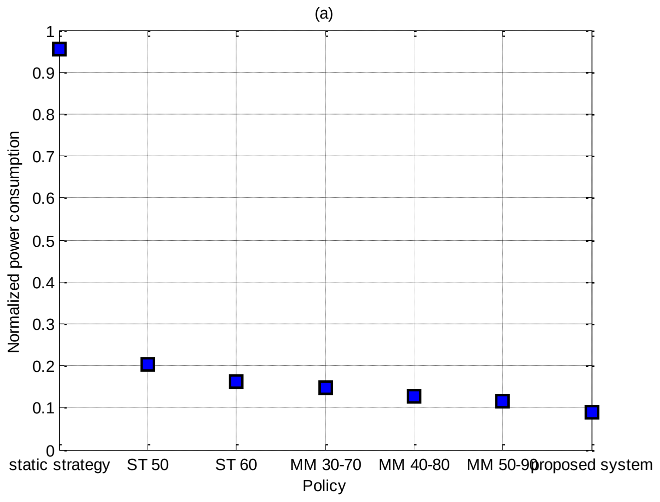
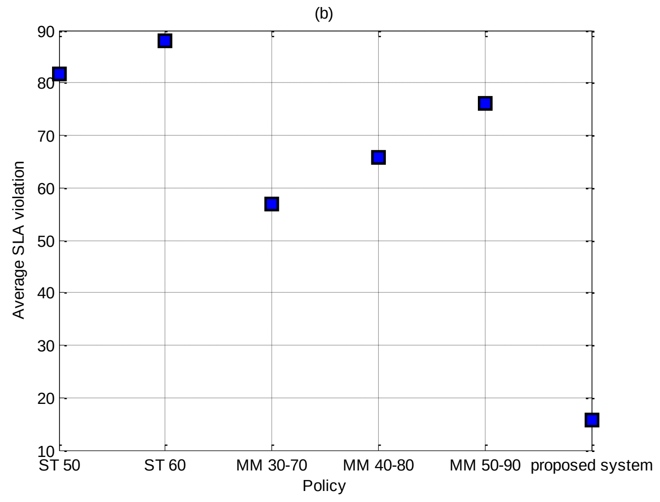


Figure 8. Normalized power consumption compared to First Fit Decreasing and Best Fit Decreasing bin backing heuristics

Figure 9 depicts a performance comparison conducted between the proposed architecture along with both the Minimization of Migration (MM) policy proposed in [16] with three representatives thresholds, and two different representatives for Migration aware single threshold policy (ST) proposed in [17]. The conducted experiments compares the performance of the proposed strategy with MM, ST policies in respect to number of migrations, SLA violations, power consumption under variable applications' workloads. It can be seen that the proposed strategy outperforms ST and MM at minimizing the number of migration. As instead of only relying on VMs migration policy for controlling the performance under volatilities or exceptional situations, the proposed strategy rather supports set of actions including: VM resizing, and DVFS methods along with migration. Furthermore, the proposed strategy shows better energy saving than ST, and MM with all their variable representatives. Meanwhile, applying the proposed strategy leads to significant decreasing at SLA violations compared to that obtained from the different representatives for ST and MM polices. Revealing that setting a fixed thresholds either for both upper utilization or lower utilization which the case in the MM policy, or setting single upper utilization as in ST policy, doesn't fit rapid and sudden fluctuations of cloud computing applications' workload especially web applications workload.





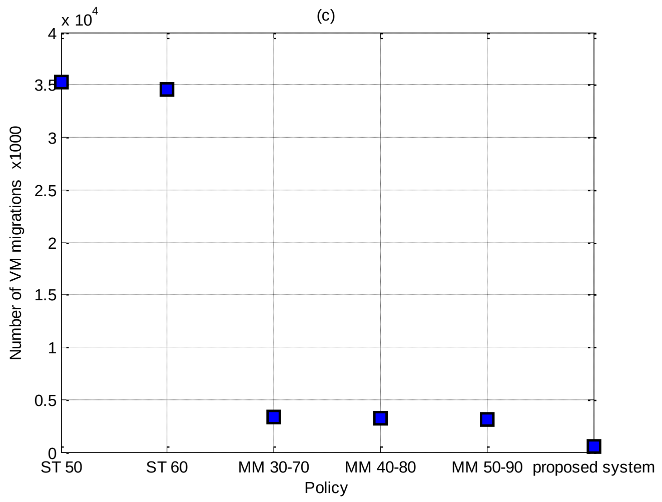


Figure 9. Proposed Architecture performance analysis: a) Normalized power consumption, b) SLA violation, c) Number of VMs' migrations

6. Conclusion

This work proposes autonomic power aware SLA-oriented cloud resources orchestration two-tier architecture. The proposed architecture geared for leveraging cloud system resources utilization, ensuring explicit guarantees on web applications' responsiveness obligations, meanwhile achieving power consumption minimization objectives. The proposed architecture consolidates heuristic methodologies along with control theory approaches in a resource orchestration hierarchical structure. Firstly, an autonomic heuristic global controller is presented. The proposed global controller exploits heuristics methodology for mapping VMs to the appreciate cloud resources in accordance to heuristic multidimensional objectives based placement strategy. Secondly, a proactive fuzzy-logic based local control is proposed. The proposed local controller aimed at in confronting workloads' sustainable fluctuations via proactive amendment for the placement and provisioning schedules. Furthermore, the proposed local controller oriented towards maintaining active power management policy especially during transient peak of usage thereby, mitigating overall costs, and extending resources capacity and performance capabilities. Simulation results and comparisons, demonstrate that the proposed architecture significantly surpasses previous approaches in terms of total energy consumption, furthermore maintaining web applications SLAs objectives despite dynamic workload scenarios.

References

[1] F. Borko and A. Escalante, (eds.), "Handbook of cloud computing", Springer, (**2010)**.
[2] Rimal, B. Prasad, E. Choi and I. Lumb, "A taxonomy and survey of cloud computing systems", Proc. of the Fifth International Joint Conference on, INC, IMS and IDC, NCM'09, IEEE, (**2009)**, pp. 44-51.
[3] F. Ian, Y. Zhao, I. Raicu and S. Lu, "Cloud computing and grid computing 360-degree compared", Proc. of Grid Computing Environments Workshop, GCE'08, IEEE, (**2008)**, pp. 1-10.
[4] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software: Practice and Experience, vol. 41, no. 1, **(2011)**, pp. 23-50.
[5] B. Rajkumar, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation computer systems, vol. 25, no. 6, (**2009)**, pp. 599-616.

[6] P. Pankesh, A. Ranabahu and A. Sheth, "Service Level Agreement in cloud computing", Cloud Workshops at OOPSLA, (**2009**).
[7] Q. Andres, H. Kim, M. Parashar, N. Gnanasambandam and N. Sharma, "Towards autonomic workload provisioning for enterprise grids and clouds", Proc. of the 10th IEEE/ACM International Conference on In Grid Computing, IEEE, (**2009**), pp. 50-57.
[8] N. Ripal and K. Schwan, "VirtualPower: coordinated power management in virtualized enterprise systems", In ACM SIGOPS Operating Systems Review, vol. 41, no. 6, ACM, (**2007**), pp. 265-278.
[9] N. Ripal, P. England, P. Sharma and A. Singh, "Feedback driven qos-aware power budgeting for virtualized servers", Proc. of the Fourth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks (FeBID), (**2009**).
[10] B. Jing, Z. Zhu, R. Tian and Q. Wang, "Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center", Proc. of IEEE 3rd International Conference on Cloud Computing (CLOUD), IEEE, (**2010**), pp. 370-377.
[11] Yusoh, Z. I. Mohd and M. Tang, "A penalty-based genetic algorithm for the composite SaaS placement problem in the cloud", Proc. of the 2010 IEEE Congress on Evolutionary Computation (CEC), IEEE, (**2010**), pp. 1-8.
[12] Z. Qian and G. Agrawal, "Resource provisioning with budget constraints for adaptive applications in cloud environments", IEEE Trans. On Services Computing, vol. 5, Issue 4, (**2010**), pp. 497 -511.
[13] I. Waheed, M. Dailey and D. Carrera, "SLA-driven adaptive resource management for web applications on a heterogeneous compute cloud", Cloud Computing, **(2009)**, pp. 243-253.
[14] T. C. Chieu, A. Mohindra, A. A. Karve and A. Segal, "Dynamic scaling of web applications in a virtualized cloud computing environment", Proc. of the IEEE International Conference on e-Business Engineering, ICEBE'09, IEEE, (**2009**), pp. 281-286.
[15] S. Shekhar, A. Kansal and F. Zhao, "Energy aware consolidation for cloud computing", Proc. of the 2008 conference on Power aware computing and systems, USENIX Association, (**2008**).
[16] B. Anton, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing", Future Generation Computer Systems, vol. 28, no. 5, (**2012**), pp. 755-768.
[17] B. Anton and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers", Proc. of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), IEEE, (**2010**), pp. 577-578.
[18] Z. Wang, Y. Chen, D. Gmach, S. Singhal, B. J. Watson, W. Rivera, X. Zhu and C. D. Hyser, "Appraise: Application-level performance management in virtualized server environments", IEEE Trans. on Network and Service Management, vol. 6, no. 4, (**2009**), pp. 240-254.
[19] Kjær, M. Ansbjerg, M. Kihl and A. Robertsson, "Resource allocation and disturbance rejection in web servers using slas and virtualized servers", IEEE Trans. on Network and Service Management, vol. 6, no. 4, (**2009**), pp. 226-239.
[20] K. H. Kim, A. Beloglazov and R. Buyya, "Power-aware provisioning of cloud resources for real-time services", Proc. of the 7th International Workshop on Middleware for Grids, Clouds and e-Science, ACM, (**2009**).
[21] G. Von Laszewski, L. Wang, A. J. Younge and X. He, "Power-aware scheduling of virtual machines in dvfs-enabled clusters", Proc. of the IEEE International Conference on Cluster Computing and Workshops, CLUSTER'09, IEEE, (**2009**), pp. 1-10.
[22] J. Chen and C. Kuo, "Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms", Proc. of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), (**2007**), pp. 28-38.
[23] V. Venkatachalam and M. Franz "Power reduction techniques for microprocessor systems", ACM Computing Surveys (CSUR), vol. 37, Issue 3, (**2005**), pp. 195-237.
[24] A. M. Elewi, M. H. Awadalla and M. I. Eladawy, "Energy Efficient Real Time Scheduling of Dependent Tasks Saring Resources", Proc. of the 2008 High Performance Computing and Simulation Conference (HPCS), (**2008**), pp. 107-116.
[25] E. M. Saad, M. H. Awadalla, M. Shalan and A. M. Elewi, "Energy-Aware Task Partitioning on Heterogeneous Multiprocessor Platforms", IJCSI International Journal of Computer Science Issues, vol. 9, Issue 2, no. 1, (**2012)** March.
[26] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim and M. Zaidi, "The Bees Algorithm–A Novel Tool for Complex Optimisation Problems", Proc. of IPROMS 2006 conference, (**2006**), pp. 454-461.
[27] X. Wang and Y. Wang, "Co-con: Coordinated control of power and application performance for virtualized server clusters", Proc. of the 17th International Workshop on Quality of Service, 2009, IWQoS, IEEE, (**2009**), pp. 1-9.

[28] D. Gmach, S. Krompass, A. Scholz, M. Wimmer and A. Kemper, "Adaptive quality of service management for enterprise services", ACM Transactions on the Web (TWEB), vol. 2, no. 1, **(2008)**, pp. 8.
[29] D. Henriksson, Y. Lu and T. Abdelzaher, "Improved prediction for web server delay control", Proc. of 16th Euromicro Conference on Real-Time Systems, ECRTS 2004, IEEE, (**2004)**, pp. 61-68.
[30] G. J. Klir and B. Yuan, "Fuzzy sets and fuzzy logic: Theory and Applications", Possibility Theory versus Probability Theory, Prentice Hall, **(1995)**, pp. 200-207.
[31] S. Seltzsam, D. Gmach, S. Krompass and A. Kemper, "Autoglobe: An automatic administration concept for service-oriented database applications", Proc. of the 22nd International Conference on Data Engineering, ICDE'06, IEEE, (**2006)**, pp. 90-90.
[32] X. Liu, J. Heo, L. Sha and X. Zhu, "Adaptive control of multi-tiered web applications using queueing predictor", Proc. of the 10th IEEE/IFIP on Network Operations and Management Symposium, NOMS 2006, IEEE, (**2006)**, pp. 106-114.
[33] Man Jr, E. G. Co, M. R. Garey and D. S. Johnson, "Approximation algorithms for bin packing: A survey", Approximation Algorithms for NP-Hard Problems **(1996)**, pp. 46-93.

Author

R. R. Darwish

He received her B.Sc., M.Sc., and PhD degrees in Electronics and Communications Engineering from Helwan University in 2000, 2004, and 2009, respectively. She is currently a lecturer at the Mechatronics Section of the Mechanical Engineering Department, Helwan University. Her research interests include image processing and wireless sensor networks, and cloud computing.