# A Survey of Resource Management in Multi-Tier Web Applications

Dong Huang, Bingsheng He, and Chunyan Miao

***Abstract*—Web applications are mostly designed with multiple tiers for flexibility and software reusability. It is difficult to model the behavior of multi-tier Web applications due to the fact that the workload is dynamic and unpredictable and the resource demand in each tier is different. Those features also cause the task of resource allocation for multi-tier Web applications very challenging. In order to meet service level agreements (SLAs) with minimal resource costs, Web service providers should dynamically allocate appropriate resources to each tier. This is particularly important to minimize the monetary cost in the pay-as-you-go cloud computing environments. Recently, a number of rule and model based approaches have been proposed for resource provisioning in cloud computing. In this survey, we identify challenges of the resource allocation problem and conduct a comparative review on those rule and model based approaches for resource allocation in multi-tier Web sites. Given the analysis on their advantages and limitations, we outline research directions to further improve the effectiveness of resource management in multi-tier Web applications.**

***Index Terms*—Web service, resource allocation, multi-tier architecture, learning, control theory.**

## I. Introduction

WITH the advancement of Web technologies, the Internet is shaping our future by providing versatile Web services, such as online trading and entertainment. Among various Web services, E-commerce is a very important type of Web service. It is viewed as a key business model of the future due to ease of transaction, quick response time and less overhead. Today, it has become a business with huge sales and revenue. The retail E-commerce sales in US for the third quarter of 2011 have reached to $48.2 billion [1]. In addition, J.P. Morgan forecasts that online retail commerce in the U.S. alone will grow 13.2 percent to $187 billion and global E-commerce revenue will hit a whopping $963 billion by 2013 [2]. The strong growth of Web services requires scaling and flexible design of the underlying system infrastructure for Web applications.

Web applications are enriched with different components including web interfaces, logics and databases. Currently, most Web applications are designed as multi-tier systems due to flexibility and software reusability [3]. In such an architecture, each tier has different functionality. For example, the 3-tier Web application architecture, which consists of presentation, application and data tiers, has been widely used [4], [5], [6], [7]. Due to different functionalities, tiers have significantly different requirements of computing resources such as CPU, main memory and disks. There is a tradeoff between the quality of service (QoS) and resource cost for the resource management in multi-tier Web applications. The service providers usually provide a certain number of QoS requirements (e.g., response time and throughput) to users, which are generally defined through service level agreements (SLAs) between a Web service provider and its users. When the QoS provided by a Web service provider satisfies the given SLA, the service provider earns revenue. Otherwise, it has to pay a penalty back to the users. Therefore, given QoS requirements, the objective of resource management in multi-tier Web applications is to allocate appropriate resources (e.g., CPU, main memory and disk bandwidth) to each tier such that the total resource cost is minimized.

Capacity planning is a classic method to determine the quantity of resources for the given QoS requirement [8]. However, capacity planning is basically a long term and almost static decision, and the resources are determined by the maximum Web application request rate in the target period to avoid excessive penalty. The maximum application request rate can be estimated according to a prediction model or historical data [9]. However, very short bursts in request rate are common in many applications [9]. The highest resource consumption at the peak load is relatively rare. For example, Fig. 1 shows the CPU and disk utilization of a production SAP application for a 24-hour period [10]. We can observe significant fluctuation in utilization caused by dynamic workloads. Moreover, the CPU utilization and the disk utilization do not demonstrated a clear strong correlation. The CPU utilization is below $50\%$ most of the time while the disk utilization is below $20\%$ around $70\%$ of the time. Most of the resources are wasted in a long duration. Therefore, the static schemes such as capacity planning are extremely inefficient and wasteful when the workload fluctuates.

Thanks to the virtualization technologies and the wide adoption of cloud computing system infrastructures, dynamic resource allocation according to the fluctuations in request rate becomes feasible in the cloud (both public and private ones). The relationship between the computing resources and the cost has been widely investigated [11], [12], [13], [14], [15]. Particularly, as the cloud computing paradigm is emerging, pay-as-you-go price schemes associate resource consumption with the monetary cost [16]. For example, Amazon charges users less than $0.1 per virtual machine hour on its small instance (or virtual machine). By using hypervisor technologies such as Xen [17] and VMware [18], the resources (e.g.,

Manuscript received March 1, 2013; revised November 26, 2013.

D. Huang is with Institute for Infocomm Research, Agency for Science Technology and Research (A*STAR), Singapore, 138632 (e-mail: huangd@i2r.a-star.edu.sg.)

B. He and C. Miao are with School of Computer Engineering, Nanyang Technological University, Singapore, 639798 (e-mail: {bshe,ASCYMiao}@ntu.edu.sg.)

Digital Object Identifier 10.1109/SURV.2014.010814.00060

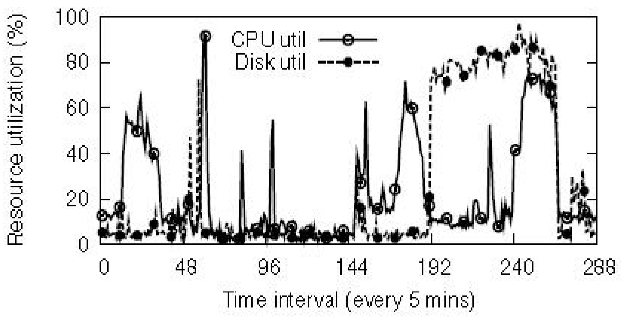

Fig. 1. Average CPU utilization and peak disk utilization in a production SAP application server for a 24-hour period [10]. We can observe highly dynamic resource utilization caused by workload dynamics.

CPU, main memory and bandwidth) can be dynamically allocated to virtual machines (VMs). In the cloud environment, the computing resources can be dynamically scaled over the Internet on demand. In Amazon, we can use virtual machines with different capabilities and/or use different number of virtual machines. Web service providers can further decrease their cost expenditure by dynamically allocating the resources among applications. Furthermore, resource management is also important for other structured or unstructured data processing systems [19], [20].

Challenges come with opportunities. There are a number of challenges on the effectiveness of resource allocation in multi-tier Web applications in the virtualized cloud environments.

- The first challenge is that different classes of resources have different impact on the QoS. For example, CPU share has larger impact on the QoS than other resources for computing intensive applications. Memory and disk can have larger impact on other cases. However, the relationship between the QoS and those resources is highly nonlinear and quite difficult to derive [21], [22].
- Second, compared with a single-tier application, the resource allocation for multi-tier applications is more difficult due to the fact that the resource demand at each tier is different. Different tiers may interact with each other, leading to different impact on the QoS. Therefore, it is difficult to determine when and how much to provision.
- Third, a coarse-grained and inappropriate approaches may cause performance degradation or low resource utilization, and finally causing penalty. Most existing works approximate the relationship based on some statistical properties of historical measurements [23], [24], [25]. Though these methods can help to increase the efficiency of resource allocation, the application performance is still not be controllable. For example, [24] approximated the relationship between the mean response time and throughput by conducting a number of simulations. However, experiment results also show that the mean response time perceived by the users can be very different for the same parameter settings.
- Fourth, cloud providers do not offer any performance guarantees with regard to the application level performance. Also, the I/O bandwidth including disk and network is usually dynamic and unpredictable [26], [27]. Different pricing or economic strategies can have huge impact on multi-tier resource management. The relationship between distributed systems and economics in cloud computing is studied in [26] and [27] used SVM to predict the interference score for different I/O workloads in order to offer better fairness for users.

We examine whether current resource allocation algorithms can address all above-mentioned challenges. Recently, a number of algorithms for the resource allocation in multi-tier Web applications have been proposed [23], [28], [29], [30], [31], [32], [33], [34], [35]. Those algorithms can be roughly divided into two categories: rule and model based approaches. Rule based approaches are mainly used for action selection. The rule based methods are basically in reinforcement learning (RL) [31], [34], [36], statistical machine learning (SML) [35], [37], and fuzzy control [38], [39], [40], [41], [42], [43], [44], [45]. Specifically, [31] studied the application of RL and ANN for multicore resource allocation. [32] formulated the virtual machine packing problem as a multi-objective optimization problem and employed a genetic algorithm (GA) to allocate the resources. Different from reinforcement learning and statistic machine learning, fuzzy control is easy to implement and efficient to handle uncertainty for the resource management in multi-tier Web sites. However, appropriate fuzzification setting is the key for a fuzzy control system. In this case, it is not easy to derive appropriate rules for the resource management. How to model the complex multi-tier Web sites accurately based on fuzzy logic is an important challenge. The major advantage of those algorithms is that they can approximate the resource demand by learning historical data without explicit model knowledge. Due to unpredictable workload and the nonlinear property of multi-tier Web application architecture, however, these algorithms cannot guarantee given QoS requirements or provide theoretical system performance analysis.

Model based approaches on the other hand, not only provide QoS guarantees, but also provide more insights into the dynamics of the multi-tier system. This will help users get understanding on how the system evolves. Currently, most model based approaches are based on control theory since it has been identified as a powerful mechanism for dealing with the uncertainty and disturbance of a system by using feedback control [46], [47]. For instance, [23] considered the problem of resource allocation on shared data centers, where they modeled a server resource as a generalized process sharing server and used a time-domain description of the server to model transient system states. [24], [28] studied the utilization of CPU in terms of mean response time based on control theory. [48] used the exponentially weighted moving average (EWMA) model to predict the demand for the number of servers. [49] developed a novel self-adaptive neural fuzzy control based server provisioning approach to provide specified delay guarantee, where the approach combined the strength of both machine learning and control-theoretic techniques. Most existing studies correct the errors based on feedback control. A distinct advantage of control theory for the system is that it can provide rigorous methodology for modeling, analysis, design and evaluation of the control system [50]. For example, stability is a very important performance metric for control systems. Lacking of stability analysis may cause large oscillation response in the control system, sometimes even causes the system to be unstable. In practice, control theory provides guidelines for choosing appropriate control parameters to ensure stability and good performance of the closed-loop system. Due to the distinct characteristics of feedback control, algorithms based on control theory have become popular for the resource allocation in multi-tier Web applications. Furthermore, they have demonstrated their effectiveness in resource management [4], [5], [21], [30], [51], [52].

In multi-tier systems, the resource management is conducted in a virtual computing environment. The resource management we consider in this article is to decide whether to accept a request or scale the computing resources at each tier in order to meet the given QoS requirement. To the best of our knowledge, this is the first survey on the resource management in multi-tier systems. Though some survey papers on resource management in computing systems have been published recently [53], [54], [55], the system model they mainly focused on is quite different from the multi-tier systems we consider in this article. For instance, [53] focused on how to achieve a balance between resource providers and consumers in grid systems based on the bargaining model when consumers compete to acquire resources. [54] reviewed the approaches in the implementation of resource management in existing grid systems. [55] surveyed various task scheduling algorithms and classified them based on various parameters, such as distributed, hierarchical and response time. Therefore, these existing survey papers do not explore the issues and approaches for the resource management in multi-tier Web sites. Still, our focus is on rule and model based resource management specifically for multi-tier Web applications. We believe that this survey is attractive to web researchers on resource management, and also to resource management researchers on the challenges from multi-tier design of web applications.

In order to provide a full understanding of existing works on resource management for the research community, we review rule and model based approaches for the resource allocation problem for multi-tier Web sites in this survey. The underlying system infrastructure hosting the Web site has the capability of dynamic resource allocation and deallocation, particularly for virtualized cloud environments. This survey is to serve those purposes: (1) providing an overview of existing works on resource allocation for multi-tier Web applications; (2) analyzing the advantages and limitations of rule and model based approaches for the problem; (3) identifying the open problems in resource management of next-generation multi-tier Web applications. The directions are mostly focused on model based approaches, since we believe they are the most promising resource management mechanism under different contexts.

The paper is organized as follows. Section II describes the multi-tier Web application architecture, and defines the resource management problem. The survey of machine learning algorithms and fuzzy control for the resource management is presented in Section III. Section IV first gives a survey of existing works on feedback control based approaches for the resource management in multi-tier Web sites, then discusses the challenges for the resource management. Finally, Section V identifies the open problems for resource management in multi-tier web sites and provides suggestions on how to improve the performance of existing works, followed by the concluding remarks in Section VI.

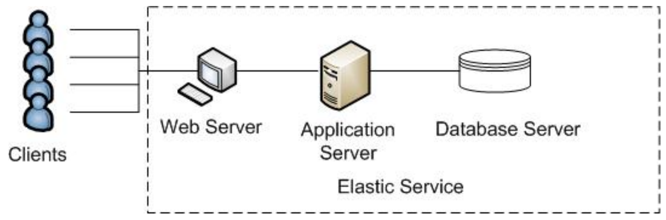

Fig. 2. A 3-Tiered Web application architecture.

## II. Resource Management for Multi-Tier Web Applications

In this section, we first describe the multi-tier Web application architecture, and then define the resource management problem.

### A. *Multi-Tier Web Application Architecture*

Though a single-tier architecture has relatively simple structure and is easy to setup, most modern Web sites use a multi-tier architecture. This architecture partitions the application process into multiple tiers. Each tier provides a certain functionality. The benefit of such an architecture is that it can provide a high level of scalability and reliability [3]. Furthermore, it is also useful for monetary cost optimizations for workflows [56]. However, the resource allocation among these tiers will be more difficult due to the interdependency between the tiers. A multi-tier Web application may span multiple nodes. Specifically, most multi-tier Web applications use a 3-tier architecture, as shown in Fig. 2. The three tiers include presentation tier, application tier and data tier, implemented as web server, application server and database server, respectively.

The first tier named presentation tier consists of Web servers. It displays what is presented to the user on the client side within their Web browsers. For the Web server tier, it mainly has three functions [5]: (1) admiting/denying requests from the clients and services static Web requests; (2) passing requests to the Application server; (3) receiving response from Application server and sends them back to the clients. Examples of web servers include Apache Server and Microsoft Internet Information Server (IIS).

The second tier named application tier consists of Application servers. Business logic processing is performed at this tier. There also are three functions at the Application server tier, which include: (1) receiving requests from the Web server; (2) looking up information in the database and processes the information; (3) passing the processed information back to the Web server. Application servers mainly use Apache Tomcat and Sun Java System Application Server.

The last tier named data tier consists of database servers. It handles database processing and data accessing. Database

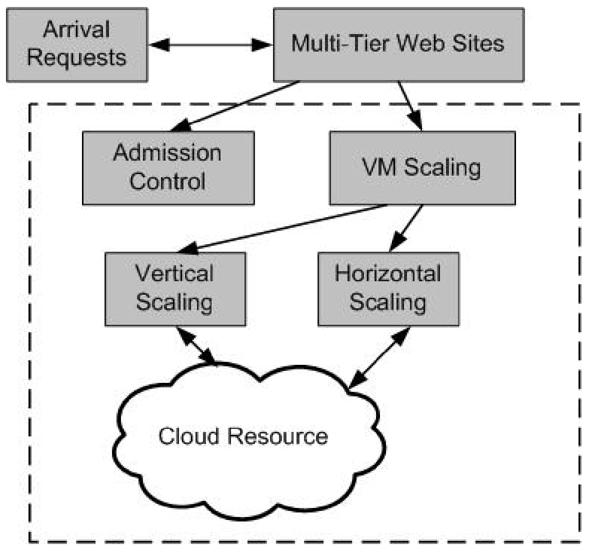

Fig. 3. Request process model for multi-tier Web applications.

server tier is used to store and retrieve a Web site's information (e.g., user accounts, catalogs to reports and customer orders).

Today, most Web applications have strict performance and QoS requirements, which are represented as SLAs. The QoS requirements can be mapped to certain computing resources. Suppose a request is proceeded under such an architecture structure. Then the resource demand in each tier will be different and cannot be characterized accurately in practice. Moreover, the three tiers have different optimizations. For example, web page caching is an effective technique to reduce the computational and memory pressures on web servers. That causes different system behavior whether the request page is in the web page cache or not. Those features cause the task of QoS guarantees not to be trivial.

## B. Problem Definition

The target of resource management in multi-tier Web applications is to allocate the computing resources to meet the resource demand with minimal resource cost. Therefore, resource management not only helps to improve the efficiency of resource utilization, but also satisfies the QoS requirements of different types of requests.

A general request processing model for multi-tier Web applications is illustrated in Fig. 3. The resource management includes two modules (i.e., admission control and VM scaling). Time is divided into a series of intervals. The intervals can be adjusted to balance the overhead of resource management and the gain from resource management. At the beginning of every interval, the admission control module is used to determine whether a request is admitted for service, while the VM scaling module is used to adjust resource usage in terms of the VM size (for VM vertical scaling) or the number of VM instances (for VM horizontal scaling).

The major functionality of the admission control module is to prevent sudden overload, which causes significant performance degradation. When providing service to clients, Web service providers need to allocate enough resources to meet the demand in order to satisfy the given QoS requirements. However, it is difficult to maintain these performance guarantees due to the fact that the workload is unpredictable.

In principle, when the workload is in heavy state, the admitted request rate should be decreased and thus the performance can be maintained. Otherwise, the requests experience long response time. An example is the "Slashdot effect" [57], which means the traffic of a Web site experiences a sudden, relatively temporary surge. However, an appropriate policy for admission control is not easy to derive. The resources (e.g., CPU and bandwidth) required by requests are different. It is difficult to model the relationship between resource demand and workloads. How to achieve a good trade-off between request dropping and request admission is a challenge for admission control. A conservative policy causes revenue loss. On the other hand, an aggressive policy causes performance degradation.

Dynamic resource provisioning is necessary, because the resource demand changes in the multiple tiers of web sites along with the workload. In the virtualized environment, VM scaling helps to improve the efficiency of resource utilization by dynamically changing the size of computing resources. Currently, VM scaling includes two types of methods. The first type is vertical scaling (i.e., scale up and down), where resources (e.g., CPU and memory) can be dynamically allocated to the same VM instance [58]. Most cloud providers including Amazon and Microsoft have offered virtual machines of different CPU capabilities and main memory capacities. Rackspace offers the functionality of runtime resizing the virtual machine [59]. This means that the resource in a single VM can change with the workloads. In this case, vertical scaling can provide fine-grained resource allocation. The second type is horizontal scaling (i.e., scale out and back), where the size of a VM is fixed. One can change the number of VM instances according to the workloads. For example, dozens of types of on demand standard VM instances varying computing capabilities and locations are available for horizontal scaling in AWS EC2 [16]. Thus, horizontal scaling provides coarse-grained resource allocation. Horizontal and vertical scaling are complementary with each other, and can be combined for resource management. Compared with traditional static method (i.e., capacity planing), the elasticity brought by VM scaling helps Web service providers to further reduce the resource cost.

Admission control and resource allocations in multiple tiers are the key issues in resource management of multi-tier web sites. We review two major kinds of resource management mechanisms namely learning and control theory in Sections III and IV, respectively.

## III. Rule Based Approaches for Resource Management

Rule based approaches have been widely in complex systems for decision making. They can help to make appropriate decisions under uncertainty. In this section, we divide rule based approaches for the resource management in multi-tier Web applications into two classes: machine learning (ML) algorithms and fuzzy control.

### A. Machine Learning Algorithms

In order to conduct the resource management efficiently, we need to derive appropriate policies for both admission control and VM scaling. Machine learning algorithms can improve an initial policy by learning from historical data. For the problem of allocating resources to multi-tier Web systems, a series of machine learning (ML) algorithms to improve the efficiency of resource management in each tier by learning from historical resource utilization and performance metrics have been investigated. In this subsection, we review works on ML algorithms for the resource management.

Specifically, because reinforcement learning offers the potential to develop optimal allocation policies without explicit model knowledge by learning from the consequences of each action, existing works on ML algorithms mainly focus on reinforcement learning [34], [60], [61], [36]. They require neither an explicit system model nor an explicit traffic model to learn.

RL refers to a learning process, where a learning agent can learn to make appropriate decisions through interactions with an external environment [62]. Specifically, beyond the learning agent and the environment, a reinforcement learning system consists of a policy, a reward function and a value function. Let $S$ be the set of environment states and $A$ be the set of actions, respectively. A policy determines the learning agent's behavior. It maps the states of the environment to the probabilities of selecting a possible action from the set of actions and is denoted by $\pi_t(s, a)$, where $s$ and $a$ can be expressed a vector with $s = [s_1,\ s_2, \ldots, s_n]$ and $a = [a_1,\ a_2, \ldots, a_m]$, respectively. Currently, a policy to choose an action at a state is mainly based on $\epsilon$-greedy method, which chooses an action to maximize immediate reward with probability $(1-\epsilon)$, while an action randomly with probability $\epsilon$. This class of method is used to balance the trade-off between exploitation and exploration. A reward function defines the objective in a reinforcement learning problem in an immediate sense. While a value function defines the objective in a reinforcement learning problem in the long term. Let $R$ be the return of the value function. There are two classes of value functions. One is with finite time step, $R = \sum_{t=0}^{N-1} r_{t+1}$, where $r_{t+1}$ denotes the immediate reward at time step $t$. The other one is with infinite time step, $R = \sum_{t=0}^{\infty} \gamma^t r_{t+1}$, where $\gamma \in [0,\ 1]$ is the discount factor. Accordingly, the value function estimation is expressed as a Q-function

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t \cdot [r_{t+1} - Q(s_t, a_t)] \tag{1}$$

for the case of finite time step and

$$\begin{aligned} Q(s_t, a_t) =& Q(s_t, a_t) + \alpha_t \cdot [r_{t+1} \\ &+ \gamma \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t)] \end{aligned} \tag{2}$$

for the case of infinite time step, respectively. Where $\alpha_t \in (0, 1]$ is the learning rate. The objective of the learning agent is to develop appropriate policies to maximize the long term reward based on iterative trial-and-error interactions [61]. Theoretically, RL can potentially learn optimal policies in dynamic environments due to the fact that the learning process is typically formulated as a finite Markov Decision Process (MDP).

Another popular machine learning algorithm is the support vector machine (SVM). It has been widely applied for different areas such as pattern recognition, classification and data mining. However, SVMs are not preferred in on-line applications since the training and testing complexity of standard SVM are $O(nm + m^3)$ and $(m)$ respectively, where $n$ is the data size and $m$ denotes the number of support vectors. On the other hand, some approximated methods have been proposed to reduce the complexity [63], [64], [65]. For example, [63] reduces the complexity to $O(nd_{\max}^2)$, where $d_{\max}$ is the number of basis functions selected. However, the mechanism of these revised versions of SVMs is based on approximation, which sacrifices the accuracy. This may be acceptable for many classification applications but unacceptable in resource management of multi-tier Web applications due to the strict response time requirement.

Recently, a few of works on machine learning algorithms have been proposed for the resource management problem [33], [34], [60], [66], [61], [67], [68], [36], [37], [69], [70], [71]. For admission control, [66] derived a complex rule set that can be used to identify the optimal configuration for unobserved workload based on machine learning algorithms. [61] applied RL to configure parameters automatically in multi-tier Web systems, where eight parameters at web tier and application tier are selected to consist of the state space. For each parameter, there are three possible actions: *increase, decrease* and *keep*. The policy is based on the $\epsilon$-greedy method. In order to suppress the poor performance due to bad initialization, they proposed an algorithm to construct different initialization policies for different scenarios. For VM scaling, [33] proposed an iterative model training technique based on artificial neural network (ANN) to predict computing resource demand in virtual environments. [34] applied RL to train nonlinear approximators (e.g., multi-layer perceptrons) instead of the lookup table for VM horizontal scaling, where the state is defined as the request arrival rate and the action is to determine the number of servers allocated. Since the state space grows exponentially with the number of parameters in practice, the authors applied a nonlinear function approximator as an external policy to avoid poor performance that would be expected during online learning. This method also helps to scale to larger state spaces. [36] presented a unified reinforcement learning methodology (URL) for autoconfiguration of VMs, which focused on the resources of CPU and memory. [60] proposed a decomposition formulation of RL for VM horizontal scaling in order to reduce the curse of dimensionality problem. [67], [68] configured resources for VM scaling via support vector machines (SVMs), which help to build model knowledge. [37] used a discrete-time Markov chain to capture the short term patterns in resource demand for VM scaling, where the resources prediction models are repeatedly updated when resource consumption patterns change. [69] modeled the virtualized storage systems by using statistical techniques. Specifically, a general heuristic search algorithm was proposed first to optimize the parameters of regression techniques, then the optimization approach is applied to create performance rules by using four regression techniques. [70] investigated the application of three machine learning techniques: SVM, ANN and LR for the modeling of a two tier TPC-W web application

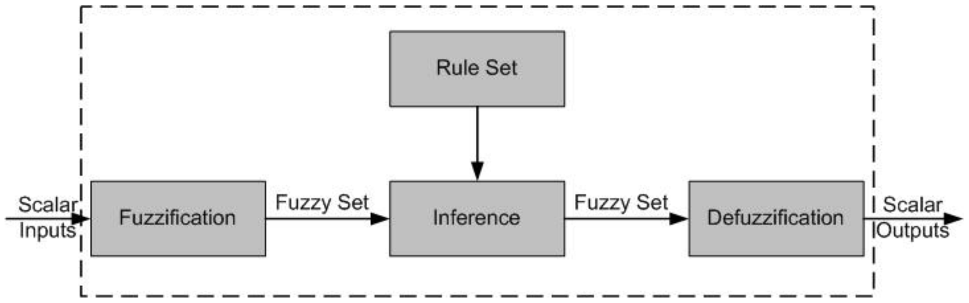

Fig. 4. A fuzzy system.

and showed that SVM provided the best prediction model. [71] proposed a multi-step-ahead load forcasting method to investigate the CPU allocation problem based on statistical learning techniques, it integrates an improved support vector regression algorithm and Kalman smoother.

However, there are several distinct limitations when applying ML algorithms for the resource management in multi-tier Web applications [68]. First, the high computational complexity in the training process cannot be ignored. For example, when RL is used, the state and action spaces scale exponentially in the number of applications, and significantly increase the overhead of trial-and-error [60]. Specifically, [61] selected eight parameters to construct the state space. Each parameter has different range and the cardinality of the state space is larger than $10^8$. In practice, there are more than one hundred configuration parameters in a multi-tier Web system. In order to reduce the search space when applying RL, we need to find an alternative technique to make the policy more efficient. The complexity reduction of RL has been widely studied [72], [73], [74]. However, these revised versions are mainly based on approximation, which sacrifice the accuracy. This is unacceptable for multi-tier Web applications.

Second, the learning speed may be unacceptable low due to strict QoS requirements in multi-tier systems and some properties in ML algorithms. Most of existing ML algorithms require a long training process in order to learn the given model accurately. For example, the performance of SVM mainly relies on the quality of the training data, while initial policy has important impact on the performance of RL. [34], [61] have demonstrated the problem of initial poor performance due to poor initial policies. Though RL theoretically obtain the optimal allocation policies, it may not converge to stationary optimal value functions within acceptable duration. Thus, given strict QoS requirements may not be satisfied.

In a word, ML algorithms can develop policies for the problem of resource allocation in multi-tier Web sites. The main advantage of ML algorithms is that it does not require much domain knowledge. However, the aforementioned disadvantages may cause the performance to be unacceptable. It is necessary to develop novel techniques to improve the performance when applying ML algorithms.

## *B. Fuzzy Control Approaches*

Fuzzy control is guided by a set of predefined rules. It seems to be a good candidate for the resource management in multi-tier systems due to easy implementation and handle. In this subsection, we review related works on fuzzy control for the resource management.

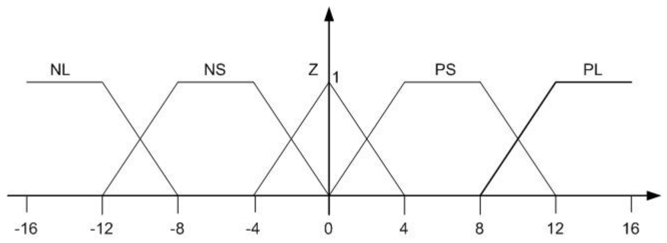

Fig. 5. Fuzzy set illustration.

In fuzzy control systems, the decision making consists of three stages: fuzzification, inference mechanism and defuzzification. A framework of a fuzzy logic system is illustrated in Fig. 4.

It can be seen that numerical variables can not be used directly in fuzzy systems. Therefore, the first stage is to fuzzify the scalar inputs. The goal of fuzzification is to convert the numeric variables into linguistic values of linguistic variables represented by fuzzy sets. Given a variable space $X$, a fuzzy set $s = (X, f)$ is denoted by a membership function $f$ defined on $X$. The membership function maps $X$ onto the interval $[0,\ 1]$, i.e., $f\ :\ X \rightarrow [0,\ 1]$. Generally, most membership functions are triangular or trapezoidal for easy implementation. Fig. 5 shows a fuzzy system, which includes five fuzzy sets: NL (negative large), NS (negative small), Z (zero), PS (positive small) and PL (positive large). When $x = 6$, it is considered as "positive small" with a degree of 1 and other fuzzy sets with a degree of 0. When $x = 2$, it is considered as "zero" with a degree of 0.5 and "positive small" with a degree of 0.5. After fuzzification, the fuzzy controller derive appropriate output in terms of linguistic variables by evaluating the fuzzy rules in the rule set. This is conducted at the Inference stage. In order to build appropriate rules to achieve ideal system performance, we need to specify the rules.

In general, the actions of a fuzzy controller depend on the predefined rules that are stored in a rulebase. These rules and actions are defined as

$$\text{IF}\, condition,\ \text{THEN}\, action. \tag{3}$$

For example, the admission control is conducted by tuning the system parameter *MaxClients* in [38]. A fuzzy rule is expressed as "IF *change-in-MaxClients* is *neglarge* and

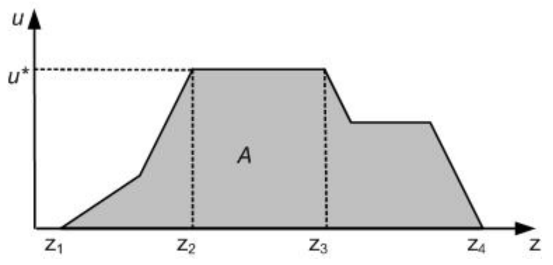

Fig. 6. Membership function for defuzzification.

*change-in-response-time* is *neglarge*, THEN *next-change-in-MaxClients* is *neglarge*." The terms *change-in-MaxClients* and *change-in-response-time* are linguistic variables. They are mapped from the numeric variables $du$ and $dt$, respectively, where $du$ denotes the change of *MaxClients* value and $dt$ is the change of the response time. *neglarge* is a linguistic value and means for negative large in size.

Finally, the output linguistic variable is converted to a numeric variable for action executing. From Fig. 4, the output at the Inference stage is fuzzy set (fuzzy rules) and can not be executed directly. In order to enable the action to be performed, these fuzzy outputs need to be converted into scalar values. Generally, the process of converting the fuzzy set is called defuzzification in fuzzy systems. There are five commonly used types of defuzzification methods: Center of area (CoA), Bisector of area (BoA), Mean of maximum (MoM), Smallest of maximum (SoM) and Largest of maximum (LoM). Suppose $\mu(z)$ is the aggregated membership function shown in Fig. 6 and $z$ is the output variable. Let $z^*$ be the defuzzified output. CoA is also known as center of gravity. In this method, $z^*$ is expressed as

$$z^* = \frac{\int_z \mu(z) z \, dz}{\int_z \mu(z) \, dz}. \tag{4}$$

For the method of BoA, $z^*$ divides the region $A$ into two sub-regions with equal area. Therefore, $z^*$ can be derived by solving the equation

$$\int_{z_1}^{z} \mu(z) \, dz = \int_{z}^{z_4} \mu(z) \, dz. \tag{5}$$

While for the method of MoM, $z^*$ is expressed as

$$z^* = \frac{\int_{z'} z \, dz}{\int_{z'} dz}, \tag{6}$$

where $z' = \{z|\mu(z) = \mu^*\}$. For the methods of SoM and LoM, $z^* = \min(\arg_z \max \mu(z))$ and $z^* = \max(\arg_z \max \mu(z))$, respectively. Especially, in Fig. 6, $z^* = z_2$ for SoM and $z^* = z_3$ for LoM. Thus, they are a natural way to handle uncertainties created by the stochastics present in most computer systems. More details about general fuzzy control can be found in [75], [76].

An example of fuzzy control effect is illustrated in Fig. 7. It can be seen that when the measured delay deviate too far away from the desired delay, a fuzzy rule will cause it back to the desired delay. For example, when the measured delay is larger than the desired delay to some degree, a fuzzy rule, which causes the measured delay to decrease, will be executed.

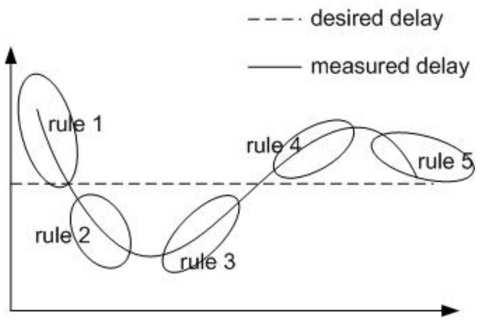

Fig. 7. Fuzzy control effect [39].

On the contrary, when the measured delay is smaller enough, a corresponding fuzzy rule will cause it to increase. Therefore, fuzzy control can ensure performance guarantees.

Recently, a few of works on fuzzy control for the resource management have been proposed in [44], [29], [38], [39], [40], [77], [78]. In [38], the admission control is conducted by fuzzy control in order to manage the QoS, where the turning parameter *Maxclients* in each interval is controlled by the fuzzy controller. For VM scaling, [29] attempted to capture the non-linear behaviors in VM resource usages by designing a fuzzy model estimator. The approache is divided into two steps. First, a fuzzy logic based modeling method is used learn the system behaviors without requiring any priori knowledge. Then a predictive controller predicts the resource demand of all VMs and takes actions based on this model. [39] proposed a neural fuzzy controller for percentile-based end-to-end delay guarantee through a virtualized multi-tier server cluster, where Gaussian membership functions are first used to fuzzify the average service time, $s_i$, and the variance of service time, $\sigma_i^2$, distribution of requests at tier $i$, respectively. Then a fuzzy neural network is applied for online learning at the Inference stage. In addition, an output scaling factor is introduced to further enhance the performance. It is model-independent and capable of adapting control parameters through fast online learning. Compared with other supervised machine learning techniques, it does not require off-line training.

[40] proposed a two-level selftuning fuzzy controller (STFC) for client-perceived end-to-end QoS guarantees. On the first level, a fuzzy controller is used to address the issue of lacking accurate server model due to the dynamics and unpredictability of pageview request traffic at the Web tier. On the second level, a scaling-factor controller is applied to compensate the effect of the process delay by adjusting the resource controller's output scaling factor according to transient server behaviors. [41] extend the work in [40] by introducing an extra self-tuning output amplification and flexible rule selection mechanism. [42] attempted to determine the number of servers to be allocated to each tier by designing a self-tuning fuzzy controller.

However, there are some issues in applying fuzzy control for the resource management. Generally, fuzzy control based approaches are model-independent and they use a set of pre-defined rules to allocate resource in multi-tier systems. There is a lack of theoretical guideline for determining appropriate values for the parameters (e.g., rule set and membership functions). In practice, it is required to conduct more simulation before designing a fuzzy system in order to achieve ideal performance. Traditional methods are based on trial and error. This strategy is effective only when the system considered is very simple. For example, [39] designed a novel method consisting of two parts (The first part is to adapt the parameters dynamically and the other part is to introduce a scaling factor to impact the rules.) to adapt parameters for the fuzzy control systems to improve the performance of resource management in multi-tier Web systems, where only the relationship between the number of servers and the response time in each tier is considered. Limiting the model for each tier, the problem becomes very simple. Obviously, this method fails to model multiple tiers as a whole and results in sub-optimal resource management. For the fuzzy controller design, how to choose appropriate values for these parameters such that the resource management is more effective is an important issue.

In summary, rule based approaches require less domain knowledge in solving the resource allocation problem in multi-tier Web sites. These approaches learn from historical data. On the other hand, they require a great deal of simulation to design appropriate rules or parameters in order to handle the resource allocation problem effectively for different scenarios. This is an important issue since there are a lot of configuration parameters in practice. In addition, there is a lack of guideline for QoS guarantees in designing rules or parameters. It causes a challenge in applying them to solve the resource allocation problem.

## IV. Model Based Approaches for Resource Management

Compared with rule based approaches, model based approaches require more domain knowledge. It may be very difficult to model a complex system based on mathematical models. However, they provide more insight into the original system and help people to understand the dynamics of the system. Currently, most model based approaches are based on control theory and queueing theory. Therefore, we further divide the model based approaches into two categories: control theory based approaches and queueing model based approaches.

In general, The system modeling is conducted based on the measurement of control inputs and control outputs. For control theory based approaches, most existing works use linear regression to model the system as a black-box problem, while the system is first simplified first and then is formulated as a queueing model in queueing theory based approaches. Compared with rule based approaches, model based approaches require more domain knowledge. For instance, when designing a feedback controller for a control system, we need to identify the system first from the viewpoint of control theory. Thus, we should identify the processing model of multi-tier Web sites first when applying feedback control for the resource management. Due to the high nonlinearity of the multi-tier Web sits, it is difficult to characterize the relationship between the workload and resource demand. For example, [79] has studied the long-term relationship between CPU utilization and entitlement and mean response time (MRT) under different workload intensities, where the request rate varies from 200 to 1000 requests/second. The results show that for the same CPU entitlement, the CPU utilization varies over different workloads. At the same time, the measured MRT also varies drastically when the workload changes. In order to provide performance guarantees for the clients with minimal resource cost, it is required to model the system accurately for resource allocation.

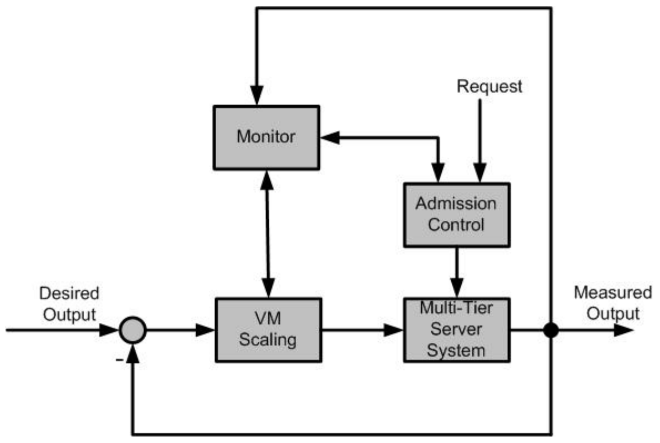

Fig. 8. Resource allocation architecture for Web service.

Note that there are errors between the approximated model and the original system no matter which model is used. In order to ensure the measured output is as close as possible to the desired output with minimal resource cost, an appropriate approach for correcting the errors due to approximation or unpredictable workload is needed. When feedback control is used to correct the errors, a modified model of resource allocation for multi-tier Web applications is illustrated in Fig. 8. The resource allocation includes three modules: monitor, admission control and VM scaling. The monitor is used to identify the relationship between control inputs and control outputs. The module consists of a workload prediction component. The performance of system identification in the monitor module is the key for allocating appropriate resources to multi-tier Web applications. Two approaches have been proposed: (1) linear regression model based approaches [80], [81], [82]. Under this model, they treat the problem as a black-box problem. Since multiple classes of resources affect a performance metric, the system is modeled as a multiple-input and multiple-output (MIMO) control system. Let $\mathbf{u}(t) = [u_1(t), u_2(t), \ldots, u_n(t)]^T$ be the input vector and $\mathbf{y}(t) = [y_1(t), y_2(t), \ldots, y_m(t)]^T$ be the output vector. Then a linear regression model is selected to model the relationship between $\mathbf{u}(t)$ and $\mathbf{y}(t)$. (2) Queueing model based approaches [4], [23], [25], [58]. By simplifying the problem, we can model the problem based on queuing theory. For example, an $M/GI/1$ (where the arrival process is memoryless, the service times of successive customers are independent, and there is a single server) Processor Sharing queue is applied to model the abstraction for the multi-tier Web application [4], [5].

According to the request process model for multi-tier Web applications described in Fig. 3, the resource management consists of two modules: admission control and VM scaling.

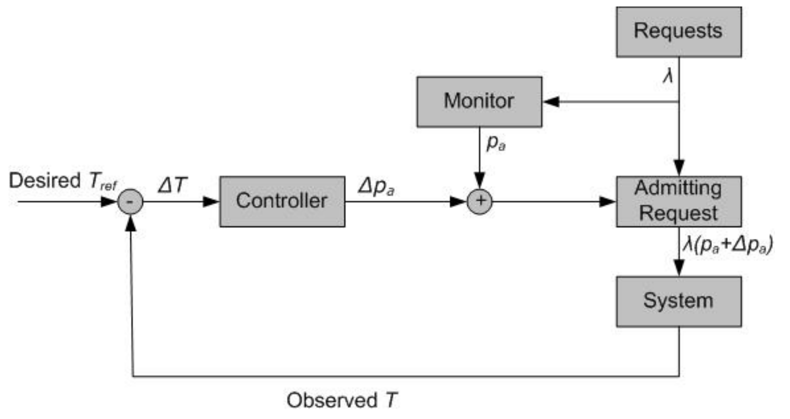

Fig. 9. Admission control model for Web service based on feedback control.

We separately survey existing works related to these two modules as follows.

## A. Admission Control

Note that the relationship between the request rate and the resource demand is nonlinear. Most existing works model the system based on queueing theory and linear regression. Due to complexity, most existing works only focus on admission control at the Web server tier [4], [5], [83], [84], [85], [86], where [4], [5], [85], [86] applied queuing theory to model the relationship between the mean response time and the request arrival rate and then conducted admission control at the Web tier. On the other side, [83], [84] conducted admission control based on linear regression. In addition, [25], [87] considered admission control at each tier by modeling the multi-tier Web applications as a closed queuing network.

*1) Queueing Theoretic Approaches::* When system identification is based on queuing theory, the multi-tier Web applications is modeled as a $M/GI/1$ Processor Sharing queue [4], [5], [88]. Let $p_a(t)$ be the admission probability for requests at $t^{th}$ interval. Let $\lambda(t)$ be the request rate. Then the effective arrival rate to the Web site is $\lambda_a(t) = \lambda(t)p_a(t)$. The mean response time $T_{PS}(t)$ is given by

$$T_{PS} = \frac{\mathrm{E}[X]}{1 - p_a(t)\lambda(t)\,\mathrm{E}[X]}, \tag{7}$$

where $\mathrm{E}[X]$ is the mean job size. Based on Eq. (7), $p_a(t)$ can be obtained by setting $T_{PS} = T_{ref}$, where $T_{ref}$ denotes the desired response time. However, the model still cannot fully capture the behavior of the multi-tier Web applications. The observed $T_{PS}$ is not equal to $T_{ref}$ in practice. In order to correct the errors $\Delta T = |T_{PS} - T_{ref}|$ effectively, [4] developed a self-tuning proportional integral (PI) controller, which updates $p_a(t)$ by $p_a(t+1) = \alpha p_a^{pred}(t+1) + (1-\alpha)p_a(t)$, where $\alpha > 0$ is constant and $p_a^{pred}(t)$ is the prediction of $p_a(t)$. While [5] introduced an adaptive controller in response to the errors, $\Delta T$, where the model parameter for characterizing the relationship between $\Delta T$ and the adjustment of admitting probability $\Delta P_a(t)$ is updated dynamically in order to ensure that the approximation error is minimized. The closed control system is illustrated in Fig. 9, where the controller is used to correct the error $\Delta T$ such that $T_{PS}$ approaches $T_{ref}$.

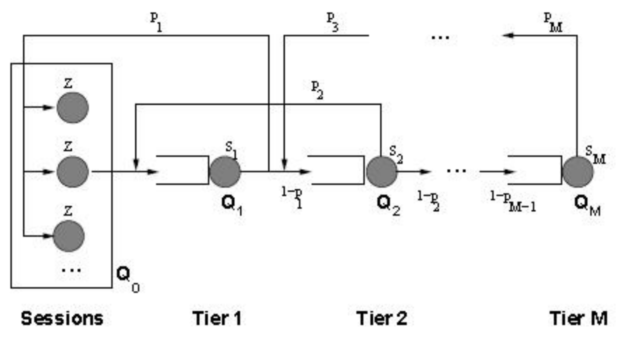

Fig. 10. Basic processing model of a multi-tier application. [25].

Note that the resource demand for a request in each tier is different. Only focus on admission control at the Web server tier cannot capture the behavior at other tiers.

[25] developed a more complex queueing model for admission control, where admission control is considered at each tier. The request process is modeled as a closed-queueing network. The workload is assumed to be session-based, where a session issues multiple requests during its lifetime. Specifically, for an application with $M$ tiers denoted by $T_1, \ldots, T_M$. Assume no tier is replicated (e.g., each tier runs on exactly one server). Then the process of an application is modeled as a network of $M$ queues, $\mathcal{Q}_1, \ldots, \mathcal{Q}_M$. A processor sharing discipline is applied to each tier. When no admission control is considered, the request processing model is illustrated in Fig. 10. For a given session, when the process of a request completes at tier $T_i$, it either returns to queue $\mathcal{Q}_{i-1}$ at tier $T_{i-1}$ with probability $p_i$ or proceeds to next queue $\mathcal{Q}_{i+1}$ at tier $T_{i+1}$ with probability $1 - p_i$. In the last queue $\mathcal{Q}_M$ at tier $T_M$, all requests return to previous queue with $p_M = 1$. When the process of a request completes, it returns to a server in $\mathcal{Q}_0$. After some *think time*, the session issues a new request for process. Thus, the model can capture the behavior of the process of a request at each tier. Under this model, the mean response time for a given number of concurrent sessions $N$ can be computed by the Mean-Value Analysis (MVA) algorithm [89], which is used for computing expected queue lengths in closed-queueing networks.

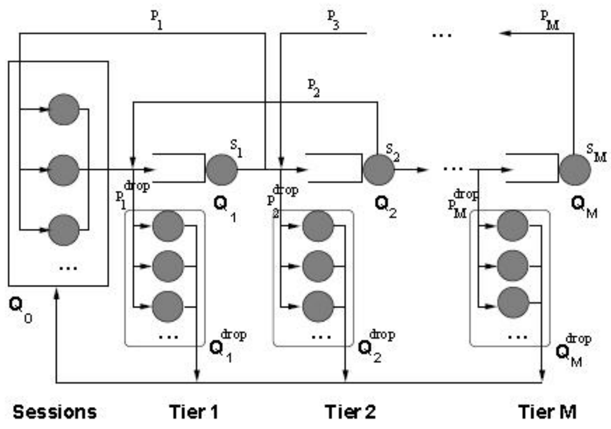

Fig. 11. Enhanced processing model of a multi-tier application. [25].

When admission control is considered at each tier, the model becomes more complex. Note that the request process capacity in each tier is different. Let $K_i$ be the concurrency limit at $\mathcal{Q}_i$ and $V_i$ be the visit ratio for Tier $T_i$. Admission control at tier $T_i$ is triggered only when the number of requests exceeds its concurrency limit. Thus, an additional transition is added to each tier. At the entrance of queue $\mathcal{Q}_i$, an infinite server queuing subsystem $\mathcal{Q}_i^{drop}$ is added. Let $p_i^{drop}$ be the probability of a request transiting from $\mathcal{Q}_{i-1}$ to $\mathcal{Q}_i^{drop}$. Then the new request processing model is illustrated in Fig. 11. In this case, the abstraction for the multi-tier Web application is assumed to be a closed-queuing network. In this queuing network, it is required to determine the drop probability $p_i^{drop}$ at each tier. The estimation of $p_i^{drop}$ at tier $T_i$ consists of two steps: (1) If there were no concurrency limits, set $p_i^{drop} = 0$; (2) Treat $\mathcal{Q}_i$ as an open, finite-buffer $M/M/1/K_i$ queue with arrival rate $\lambda V_i$. Then $p_i^{drop}$ can be estimated as the probability of buffer overflow in the $M/M/1/K_i$ queue [90].

*2) Control Theoretic Approaches::* When linear regression model is considered for system identification, most existing works focus on the admission control at Web server tier [81], [82], [91], [92], [93], [94]. Generally, a function is chosen for characterizing the relationship between the observed output and the control inputs based on a set of experiment results. Then the control parameters for the feedback control can be determined based on the derived function. Specifically, auto-regressive moving-average (ARMA) model has been applied to model the system widely. Where the relationship of the observed output $y(k)$ and the control input vector $\mathbf{u}(k)$ is expressed as

$$y(k) = \sum_{i=1}^{m} a_i(k)y(k-i) + \sum_{j=0}^{n} \mathbf{b}_j^T(k)\mathbf{u}(k-j), \qquad (8)$$

where $a_i(k)$ and $\mathbf{b}_j(k)$ are constants for estimation. In order to estimate parameters $a_i(k)$ and $\mathbf{b}_j(k)$, least-squares based methods [95] in the Matlab System ID Toolbox [96] are used to fit the input-output data collected from existing results. For example, in [91], the observed output is defined as CPU utilization, which is denoted by $y(k)$. The control input is defined as the maximum number of connections that the server permits, which is represented by $u(k)$. The objective is to maintain the system performance by controlling $u(k)$.

In summary, when applying control theory to handle the admission control problem, we can use both queueing theory and linear regression to model the system. It is important to note that much domain knowledge is required when queueing theory chosen. For linear regression approaches, how to define the relationship among the control variables is a challenge in order to achieve effective performance. In this case, we need to investigate the relationship among the control variables before applying linear regression.

## B. VM Scaling

VM scaling is a concept of dynamically allocating the computing resource for virtual machines. Some cloud providers already support VM scaling. For example, Rackspace allows users to increase the memory capacity of their VMs dynamically. In general, the goal is to minimize the resource cost, while its constraints include system capacity and QoS requirement. Due to the reduction in operation cost, a lot of VM scaling technologies have been developed [17], [18], [97]. VM scaling consists of two methods: vertical scaling and horizontal scaling according the discussion in Section II. Vertical scaling is a fine-granularity control. It is conducted by changing the size of a VM instance; while horizontal scaling is a coarse-granularity control. It is conducted to meet the demand by launching more VM instances. We survey related works as follows.

*1) Vertical Scaling:* For vertical scaling, the key is to determine appropriate resources in each VM instance in order to meet the clients' QoS requirements. Most existing works focus on two types of methods (i.e., linear regression and queuing theory) for modeling the relationship between the workload and resource demand. Compared with queuing model, linear regression model is easier to derive due to less domain knowledge required. Thus, most of existing works focus on linear regression. Specifically, the CPU demand has been investigated based on linear regression in [6], [79], [80], [98], [99], [100], [101], [102], [103] and based on queuing theory in [7], [58], [104], respectively. For example, based on the ARMA model shown as Eq. (8), [79] investigated the relationship between CPU entitlement and the inverse of MRT for vertical scaling. The vertical scaling at the bottleneck tier is shown to be more effective in [7] based on queuing theory. In addition, [10], [91], [100] have presented the analysis of multiple types of resources allocation for vertical scaling. For example, [10] investigated the combination of CPU and disk scaling based on linear regression. When linear regression model is considered, most existing works apply the ARMA model to approximate the original systems due to easy implementation [105]. Since the ARMA does not require explicit domain knowledge, the modeling errors may be large and cause performance degradation. This is an important issue for the performance improvement.

a.) *Queueing Theoretic Approaches:*

When queuing model is considered for system identification, the request process for the multi-tier Web applications can be modeled as an $M/GI/1$ Processor Sharing queue

($M/GI/1/PS$) [7]. Due to intensive domain knowledge required, few works focuses on queuing model for VM vertical scaling. Recently, [58] developed a simplified model, where the request process is modeled as a $M/G/1/PS$ queuing network. In the queuing model, consider a multi-tier application consisting of $N$ tiers. Assume there are $\Omega$ transaction types of the workload. Let $\lambda_\omega$ be the arrival rate of the workload for the transaction type $\omega$. Then the aggregate rate of the transaction as $\lambda = \sum_{i=1}^{\Omega} \lambda_\omega$. Let $u_n$ and $c_n$ be the CPU entitlement and CPU usage at tier $n$, respectively. The resident time (wait time + service time) on each tier is composed of two parts, the resident time on CPU resources and that on non-CPU resources. Let $T_{cpu}$ be the total resident time on CPU across all the tiers. Note that the CPU resident time in the $n-$th tier is represented by $T_n = \frac{r_n}{\lambda(1-r_n)}$, where $r_n = \frac{c_n}{u_n}$ is the CPU utilization of tier $n$. Then the total resident time on CPU is

$$T_{cpu} = \sum_{n=1}^{N} T_n = \sum_{n=1}^{N} \frac{r_n}{\lambda(1-r_n)} = \sum_{n=1}^{N} \frac{c_n}{\lambda(u_n - c_n)}. \quad (9)$$

The mean resident time on non-CPU resources is approximated by

$$T_{others} = \sum_{\omega=1}^{\Omega} \alpha_\omega \frac{\lambda_\omega}{\lambda}, \quad (10)$$

where $\alpha_\omega$ denotes service times of transaction type $\omega$ on all non-CPU resources of all tiers on the execution path of that transaction type. If we denote $T_{others} = \beta$ as a constant, then we have

$$RTT = T_{cpu} + T_{others} = \frac{1}{\lambda} \sum_{n=1}^{N} \frac{c_n}{u_n - c_n} + \beta. \quad (11)$$

Thus, we can approximate the relationship between $u_n$ and the given $RTT$ target from Eq. (11). According to this equation, we can further design a feedback controller to enhance the system performance.

b.) *Control Theoretic Approaches:*

When the multi-tier system is modeled as a MIMO control system, it can be characterized as a MIMO model similar to the control system in Eq. (8). Specifically, the inputs to the system are the computing resources allocated to each tier. The outputs are the measured performance metrics. For such a MIMO control system, the first step is to identify the control parameters. Then the controller design is derived based on specific objective. Clearly, the first step is the key of the vertical scaling problem. Most of existing works applied machine learning techniques on this step. For example, [79], [80] estimated the parameters by using least-squares based methods. [101] applied SVM to update the parameters. While [103] employed fuzzy rules to characterize the control model.

*2) Horizontal Scaling:* For horizontal scaling, the resource allocation means VM instance allocation. Compared with vertical scaling, the performance of horizontal scaling is more difficult to control than that in vertical scaling. In practice, it takes several minutes to start up or shut down a VM instance [106]. Inappropriate control policy may cause oscillation response in the system. In addition, there are many different VM instances for scaling [16]. In each control interval, it is required to choose appropriate type of VM instances with appropriate amount for scaling.

Horizontal scaling is suitable for distributed Web sites. For example, the database layers can run on multiple machines. The horizontal scaling usually does not affect the service. In contrast, the server resources (e.g., CPU and disk) are required to be located at the same physical machine when vertical scaling is conducted.

There are two issues in horizontal scaling. First, how many VM instances should be selected for scaling when horizontal scaling is triggered? Second, how to determine the appropriate threshold in order to ensure the performance satisfy the given requirement? Recently, there are some studies on horizontal scaling based on auto-scaling methods. Let's discuss these two issues in details.

**Determining the number of VMs:** Most related works used a set of rules to solve the problem [107], [108], where a threshold is defined for triggering to add or delete servers in order to maintain the performance guarantees. These threshold-based algorithms are simple and easy to implement. For example, one can set a threshold to add new Amazon EC2 instances in increments of 3 instances to the Auto Scaling Group when the average CPU utilization goes above 70% [107]. However, the task of choosing appropriate threshold values for these algorithms is not trivial. The scaling performance is an important issue for these algorithms. For example, a cluster with small size can increase its capacity drastically even when a VM instance is added. However, for a cluster with large size, there is not obvious change of capacity even when ten VM instances are added [109].

In order to determine the number of VM instance for scaling, the relationship between the workload and the VM instance demand is developed based on queuing theory in [110]. For a multi-tier application consisting of $k$ tiers. Let $R$ be the desired end-to-end response time. Assume it is broken down into per-tier response times, denoted by $d_1, d_2, \ldots, d_k$, such that $\sum_{i=1}^{k} d_i = R$. Let $s_i$ and $\lambda_i$ be the average service time for a request and the request arrival rate at tier $i$, respectively. Each tier is modeled as a $G/G/1$ system. Then the behavior of a $G/G/1$ system can be captured by

$$\lambda_i \geq \left[ s_i + \frac{\sigma_a^2 + \sigma_b^2}{2 \cdot (d_i - s_i)} \right]^{-1}, \quad (12)$$

where $\sigma_a^2$ and $\sigma_b^2$ are the variance of inter-arrival time and the variance of service time, respectively. Eq. (12) provides a lower bound on the request arrival rate $\lambda_i$ that can be serviced by a single server. Assume an average regression session think-time is $Z$. Then a session issues requests at a rate of $\frac{1}{Z}$. Assume the session arrival rate is $\lambda$ and the session duration is $\tau$. Then the request arrival rate is $\frac{\lambda\tau}{Z}$. Let $\eta_i$ be the number of servers needed at tier $i$. Once the capacity of a single server $\lambda_i$ has been computed, $\eta_i$ can be estimated by

$$\eta_i = \left\lceil \frac{\beta_i \lambda \tau}{\lambda_i Z} \right\rceil, \quad (13)$$

where $\beta_i$ is a tier-specific constant. Thus, Eq. (13) gives the estimation of number of servers needed at each tier, which responds to the request arrival rate, $\lambda$. While the relationship between the number of VM instances and the response time is modeled as a $M/M/m$ queueing system in [111]. On the other hand, a variant of exponentially weighted moving average (EWMA) load predictor is used in [112]. The load predictor formula is expressed as

$$E(t) = \alpha \cdot E(t-1) + (1-\alpha) \cdot O(t), \quad 0 \leq \alpha \leq 1 \tag{14}$$

where $E(t)$ and $O(t)$ represent the estimated and observed number of required VMs at time $t$. $\alpha$ is a parameter for the tradeoff between stability and responsiveness.

**Determining the threshold:** In [110], let $\lambda_{pred}(t)$ and $\lambda_{obs}(t)$ be the predicted arrival rate and the actual arrival rate during $t^{th}$ interval. Then the VM horizontal scaling is triggered if $\frac{\lambda_{obs}(t)}{\lambda_{pred}(t)} > \tau_h$ or $\frac{\lambda_{obs}(t)}{\lambda_{pred}(t)} < \tau_l$, where $\tau_h$ and $\tau_l$ are application-defined thresholds. However, the rule for threshold selection has not been investigated in these previous works yet. In practice, inappropriate threshold settings may cause oscillation response in the system [109]. For example, due to inappropriate threshold setting, a controller may launch a new VM instance in current interval due to heavy workload predicted. Later it may shut down an existing VM instance before the new VM instance is ready to use due to light workload predicted. It wastes the money as well as the non-ignorable VM acquisition.

In order to maintain the scaling performance and suppress the oscillation, a novel feedback controller based on proportional thresholding is proposed to achieve stable VM scaling in [109], where the rule for threshold selection is investigated. Let $u_{k+1}$ be the new actuator value and $u_k$ be the current actuator value. Let $y_{ref}$ and $y_k$ be the desired output and the observed output, respectively. Ideally, if the startup time of launching a VM instance is negligible, an integral controller

$$u_{k+1} = u_k + K_i \cdot (y_{ref} - y_k) \tag{15}$$

can ensure that $y_k$ is as close as possible to the desired $y_{ref}$. However, the startup and shut down time of a VM instance are not negligible and have significant impact on the scaling performance. Though a VM instance can be scaled out at any time, it is not ready to use immediately. Note that it takes several minutes to start up or shut down a VM instance. The controller in (15) cannot be directly applied for VM horizontal scaling. In order to ensure the system is stable and the desired performance is achieved, an integral controller is defined as

$$u_{k+1} = \begin{cases} u_k + K_i \cdot (y_h - y_k) & \text{if } y_h < y_k \\ u_k + K_i \cdot (y_l - y_k) & \text{if } y_l > y_k \\ u_k & \text{otherwise} \end{cases} \tag{16}$$

where $y_h$ and $y_l$ are high and low target sensor measurement. The controller will not change the control input until the sensor measurement is outside the target range. Specifically, the value of the control input increases only when the observed output is above the high target and decreases when it is below the low target. [113] developed similar policies to achieve elastic control of the storage tier. The above mentioned works are compiled into a survey table illustrated in Table I. Compared with queuing model, linear regression model is easier to implement and less domain knowledge required. It can be seen that most of existing works apply linear regression methods to model the system.

Compared with rule based algorithms, model based approaches not only can provide performance guarantees, but also ensure the closed-loop control system is controllable by tuning appropriate control parameters. The summary comparison is illustrated in Table II. It can be seen that control theoretic approaches are more appropriate for the QoS performance control in multi-tier systems.

In addition, a comparison of fuzzy control and control theoretic approaches for the resource management in multi-tier systems is also illustrated in Table II. Note that the system is very complex with high uncertainty. RL can learn the system through error and trial and thus adapts to environment. However, the learning process of RL requires a long duration in order to approximate it with high accuracy. Moreover, as aforementioned, the algorithm complexity of RL is very high. This is unacceptable for multi-tier Web applications due to the strict QoS requirement. Fuzzy control based approaches, on the other hand, learn from historical data to derive rules for the resource management problem and update rules according to change of the system. Compared with RL, they require less learning complexity and provide QoS requirement. However, the rule design is very challenge. Simple rules are easy to derive, but they may not allocate the resource efficiently. Complex rules on the other hand, may cause the system to oscillate, which leads the system to be unstable. Control theoretic approaches develop controllers from the viewpoint of control theory. Such models not only provide guideline for parameter design for given QoS requirement, but also ensure the system is stable. These advantages make control theoretic approaches more appropriate for the resource management in multi-tier Web applications.

In summary, designing an appropriate scheme for the resource management in multi-tier Web systems is an important issue in practice. Many existing works have demonstrated the effectiveness and efficiency of model based approaches in addressing the issue. They can overcome the drawbacks of other existing works. Moreover, they not only provide QoS guarantees, but also ensure system stability. Due to these distinct advantages, model based approaches are becoming more popular for the resource management in multi-tier Web systems.

## V. Open Problems and Suggestions

We have theoretically analyzed how the approaches work and discussed their respective advantages and disadvantages. Many computing resource allocation models have been investigated under different virtualized environments. Most of the surveyed models and techniques not only applicable to multi-tier applications, but also other applications involving provisioning on multiple tiers such as eScience and big data applications. The performance of these approaches has been evaluated in respective papers. On the other hand, we acknowledge that current related methods are from academia. It is unclear whether and how industry companies can adopt those methods to improve the effectiveness of resource provisioning. Thus, more benchmark studies and application development experiences will be valuable for this research field.

Having reviewed the existing studies, we have identified a number of important open problems. We divide them into

TABLE I
CATEGORIZATION OF MENTIONED WORK.

| Resource Control | Monitor | | Multiple Tiers | |
|---|---|---|---|---|
| | Queueing Theory | Linear Regression | One Tier | Every Tier |
| Admission Control | [4], [5], [25], [87], [88] | [83], [81], [82], [91], [92], [93], [94] | [4], [5], [83], [81], [82], [88], [91], [92], [93], [94] | [25], [87] |
| Vertical Scaling | [7], [58] | [6], [10], [98], [100], [102] | [100] | [7], [58], [6], [10], [98], [102] |
| Horizontal Scaling | [110] | [109], [113] | [113] | [109], [110] |

TABLE II
PROPERTY COMPARISON OF RL, FUZZY CONTROL AND CONTROL THEORETIC APPROACHES (✓: YES ×: NO).

| Approaches | RL | Fuzzy control | Control theoretic |
|---|---|---|---|
| Provide QoS guarantees | × | ✓ | ✓ |
| Provide guideline for parameter design | × | × | ✓ |
| Less learning complexity | × | ✓ | ✓ |
| Adapt to the environment | ✓ | ✓ | ✓ |
| System stability | × | × | ✓ |

two categories. The first category is on the open issues in improving current model based approaches, and the second category is on the new developing trends from cloud computing, web applications and control theory. Then we provide some suggestions in this section on how to improve the performance of existing works.

## A. Open Problems in Model Based Approaches

Despite many existing studies that we have reviewed in the previous sections, there are still some important open issues for resource allocations for multi-tier web sites hosted in the cloud environment.

We have identified the following challenges on developing model based approach for resource allocations.

First, when system identification is conducted at the monitor module, which model is appropriate to use? Though different types of models (e.g., queuing model, linear regression model and machine learning model) have been proposed, there is a lack of theoretic analysis on the advantages and limitations of these models. There is not a clear guidance on choosing the appropriate model given certain web workloads and cloud environment specifications.

Second, how to choose the set of appropriate control inputs given certain target metrics (output)? In practice, the relationship between performance metrics (e.g., mean response time and throughput) and control inputs (e.g., CPU, memory and bandwidth) can vary under different workload regions. Thus, the relationship is difficult to identify due to high complexity of the computing system and dynamic workloads. For a control output, a control input may affect it significantly under some workload regions, but has little impact on the output under other workload regions. For example, there exists a linear mapping from the CPU entitlement to $1/\text{MRT}$ within the overload region, while the MRT becomes independent of the CPU entitlement setting when the system is underloaded [79]. The MRT is uncontrollable when only using CPU entitlement in this region. A lack of analysis of the impact of control inputs on the control output under different workload regions may cause a resource allocation scheme to be ineffective under some workload regions.

Third, how to determine the feasibility of control inputs? When control theory is applied to computing systems, control variables must be defined in a meaningful way [82]. Under a given control system, for an output reference value, the corresponding control inputs may be unfeasible in computing systems. For example, under some resource allocation rules, the derived value of a control input is negative, indicating the unfeasible input.

Finally, how to ensure the controller is robust to the changing of the workload [79]? A controller with robustness ensures the control system is stable and works well under different workload regions. Most existing works on feedback control for the resource management only focus on the chosen system identification model and ignore the error introduced by the system identification model. In practice, the error also has significant impact on the system performance. In the worst case, the error may be quite large and causes the control system to be unstable. Therefore, this issue is very important for performance guarantees.

## B. New Trends

There are a few new trends that impact the current research of the resource allocation problem in multi-tier web applications.

**Non-linear control.** Many of the reviewed studies are based on linear control methods, which rely on the key assumption of small range operation for the linear model to be valid. However, we have observed that the QoS target settings usually have an non-linear relationship with the control inputs. It is rather questionable how existing studies perform in the real setting of dynamic workloads.

There have been extensive research efforts in nonlinear control [114], [115]. Compared with linear control, nonlinear control has several advantages, such as simpler implementation, faster response and less controlling cost. Advances have been made in the areas for non-linear control including feedback linearization, sliding control, and nonlinear adaptation techniques. Those techniques should be revisited under the context of resource allocation for multi-tier web. Compared with linear control based approaches, non-linear control based approaches can model the system more accurate due to the high nonlinearity of it. In this case, they can achieve the same QoS requirement with a smaller amount of resources. However, the major disadvantage of nonlinear control is nonlinear control requires rigorous mathematical analysis for the controller design. Researchers should address those disadvantage for applying nonlinear control. With the advance of nonlinear control, the controller design based on nonlinear control will help to improve the performance of resource management for multi-tier Web applications.

**The evolution of cloud computing.** Traditionally, service providers are required to statically buy enough computing hardware in order to meet the peak load. However, the utilization can be low due to the workload fluctuation. Recently, cloud computing becomes a new and attractive paradigm for hosting multi-tier Web applications, because of its pay-as-you-go price schemes. When multi-tier applications run in the cloud environment, the computing resource can be dynamically scaled according to the demand (i.e., elasticity).

Cloud computing has evolved into a main-stream information technology infrastructures for many applications. Driven by the wide adoption, developers and researchers have developed many useful tools for resource management in the cloud, for example, Amazon auto-scaling [107] and domain specific auto-scaling [116]. We expect that more tools and systems with advanced techniques will improve the effectiveness of resource management. Since those tools and systems may not be specially designed for multi-tier web applications, we need to examine how those techniques can be adapted to the scenario of multiple tiers.

Virtualization is the core technique for cloud computing. Though virtualization technologies have been shown to be effective tools for the resource management in multi-tier Web sites, resource allocation in multi-tier Web applications is not a straightforward job due to high complexity in computing systems and different CPUs/disks virtualization interference due to resource sharing among multiple virtual machines. The key of effective resource management is to identify the relationship between the resource demand and the performance metrics under different workloads accurately.

Beyond technological innovations, the cloud itself is evolving. One particular change is the price structure of the cloud. For example, recently Amazon EC2 has adopted spot price. The price of a certain VM type is determined by the runtime dynamics of demand and supply. That means, not every unit of resource consumption has the same price, depending on the time that we use the spot instance. When the price is very high, the system should reject more requests. Otherwise, it should admit more request when the price is very low. The resource management problem becomes more complicated, which becomes a multi-objective optimization problem. Moreover, the distinct feature of spot instances is their out-of-bid feature, when the bidding price is lower than the spot price, the allocated VM is terminated. This poses significant challenges in achieve SLA of web applications.

**The evolution of Web service.** The popularity of new web services like social networks and search engines already shift the mostly static pages into rich Internet application (RIA) technologies such as Adobe Flash. One of the major technologies is HTML5. The core aims of HTML5 are to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices. As 30 September 2011, 34 of the world's top 100 Web sites were using HTML5 - the adoption led by search engines and social networks.

In general, the resources of Web browsers include CPU, memory, or network bandwidth for web applications and devices such as GPS, cameras and microphones. The resource management includes resource access control and resource sharing. It has significant impact on the performance of multi-tier Web applications. Though today's web browsers have evolved into multi-principal operating environments, no effective method for the resource management of Web browsers has been developed [117], which leads them to be less robust than other desktop applications. Now the evolution of Web service, especially the HTML5, causes the resource demand to become diversified. This makes the resource management in Web service more complicated and challenging. In this case, a novel resource allocation scheme with the ability adapt to such a complicated environment is required. The responsive control systems become very attractive in this scenario.

In summary, the above trends bring a new dimension of resource management in multi-tier Web applications, they also cause the resource management problem more complicate. Therefore, on one hand, a further study on the problem based on existing works will continuously improve both the effectiveness and efficiency of current systems. On the other hand, the resource demand model in multi-tier systems keeps evolving, which requires us to make a revision of existing works before applying them.

## C. Suggestions for Performance Improvement

The two classes of approaches have been applied for the resource allocation problems in multi-tier systems. The effectiveness of them has been evaluated under specified models. As aforementioned discussion, there are still many challenges in the application of these approaches in multi-tier systems. In this subsection, we give some suggestions on how to improve the performance of existing works.

Note that both rule based approaches and model based approaches cannot accurately model the multi-tier systems due to the high complexity and the dynamics in the systems. One needs to set some parameters such that they are conservative in order to satisfy the strict QoS requirements. Clearly, if an approach is aggressive, it will not meet the specified QoS requirements. Thus, the difference among these existing works is the degree of conservatism. It is important to note that the objective of applying these approaches is to meet the given QoS requirement with minimum resource cost. A feasible way to improve the performance of existing works is to model the system more accurately. In order to do that, we need to identify which classes of parameters cause the given approach to be conservative or aggressive. Once the measured metrics show the current setting is too conservative, we should make some changes to these parameters such that the given approach becomes less conservative. Otherwise, we should make it becomes more conservative. Under such a policy, we can continue to improve the modeling accuracy and thus reduce more resource cost while satisfy the specified QoS requirements.

## VI. Concluding Remarks

Performance guarantee is an important issue for multi-tier Web applications. In the paper, we survey the resource allocation mechanisms including rule and model based approaches for the resource allocation in multi-tier Web applications. Rule based approaches require less domain knowledge and mathematical foundations. These approaches derive rules from historical data. They require a great deal of simulations and training data to derive appropriate rules or parameters in order to handle the resource management problem effectively for different scenarios. In comparison, model based approaches attempt to model the original system with mathematical models. This requires much domain knowledge to model the system. However, they provide more insights into the system and achieve better trade-off between QoS requirement and resources. We identify challenges of the resource allocation problem and theoretically illustrate why control theory is appropriate for the problem. Existing works have shown that feedback control based approaches are effective for the resource management in multi-tier Web applications in that control theory provides rigorous methodology for modeling, analysis, design and evaluation of the control system. Compared with the learning approaches, fuzzy control and control theoretic have the advantages of providing QoS guarantees and less leaning complexity. Compared with fuzzy control, the control theoretic approach has the further advantage of system stability and offering the guideline for parameter design. Finally, we have identified the open problems in next-generation multi-tier web site in the cloud environment, which calls for action from the community of web, control and cloud systems.

## Acknowledgements

The work was supported by SUG Grant M4080102.020 of Nanyang Technological University, Singapore.

## References

[1] US. Census Bureau. http://www.census.gov/retail/, 2011.

[2] J.P. Morgan, "Global E-Commerce Revenue To Grow By 19 Percent In 2011 To $680B." http://techcrunch.com/2011/01/03/j-p-morgan-global-e-commerce-revenue-to-grow-by-19-percent-in-2011-to-680b/, 2011.

[3] D. Barry, *Web Services and Service-Oriented Architecture: The Savvy Manager's Guide*. Morgan Kaufmann Pub, 2003.

[4] A. Kamra, V. Misra, and E. Nahum, "Yaksha: A Self-Tuning Controller for Managing the Performance of 3-Tiered Web Sites," in *12th IEEE Int. Workshop on Quality of Service*, pp. 47–56, 2004.

[5] X. Liu, J. Heo, L. Sha, and X. Zhu, "Adaptive Control of Multi-Tiered Web Applications Using Queueing Predictor," in *10th IEEE Netw. Operations and Management Symp.*, pp. 106–114, 2006.

[6] X. Wang, Z. Du, Y. Chen, and S. Li, "Virtualization-Based Autonomic Resource Management for Multi-Tier Web Applications in Shared Data Center," *J. of Systems and Software*, vol. 81, no. 9, pp. 1591–1608, 2008.

[7] Y. Diao, J. Hellerstein, S. Parekh, H. Shaikh, and M. Surendra, "Controlling Quality of Service in Multi-Tier Web Applications," in *26th IEEE Int. Conf. on Distrib. Computing Systems*, pp. 25–25, 2006.

[8] D. Menasce and V. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods*. Prentice Hall, 2002.

[9] M. Arlitt and T. Jin, "A Workload Characterization Study of the 1998 World Cup Web Site," *IEEE Network*, vol. 14, no. 3, pp. 30–37, 2000.

[10] P. Padala, K. Hou, K. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, "Automated Control of Multiple Virtualized Resources," in *Proc. 4th ACM European Conf. on Computer Systems*, pp. 13–26, 2009.

[11] P. Upadhyaya, M. Balazinska, and D. Suciu, "How to price shared optimizations in the cloud," *Proc. of the VLDB Endowment*, vol. 5, no. 6, pp. 562–573, 2012.

[12] H. Xu and B. Li, "Maximizing revenue with dynamic cloud pricing: The infinite horizon case," in *IEEE Int. Conf. Commun.*, pp. 2929–2933, 2012.

[13] A. Gandhi, Y. Chen, D. Gmach, M. Arlitt, and M. Marwah, "Minimizing data center sla violations and power consumption via hybrid resource provisioning," in *IEEE Int. Green Computing Conf. and Workshops*, pp. 1–8, 2011.

[14] S. Ren, C. Lan, and M. van der Schaar, "Energy-efficient design of real-time stream mining systems," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 3592–3596, 2013.

[15] Y. Wang, S. Chen, and M. Pedram, "Service level agreement-based joint application environment assignment and resource allocation in cloud computing systems," in *IEEE Green Technol. Conf.*, pp. 167–174, 2013.

[16] Amazon, "Amazon EC2 Instance Types." http://aws.amazon.com/ec2/instance-types/, 2012.

[17] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," in *Proc. 19th ACM Symp. on Operating Systems Principles*, pp. 164–177, 2003.

[18] M. Rosenblum, "Virtual Platform: A Virtual Machine Monitor for Commodity PCs," in *Hot Chips*, vol. 11, 1999.

[19] B. He, M. Yang, Z. Guo, R. Chen, B. Su, W. Lin, and L. Zhou, "Comet: batched stream processing for data intensive distributed computing," in *Proc. 1st ACM symp. on Cloud computing*, pp. 63–74, 2010.

[20] R. Chen, M. Yang, X. Weng, B. Choi, B. He, and X. Li, "Improving large graph processing on partitioned graphs in the cloud," in *Proc. 3rd ACM Symp. on Cloud Computing*, pp. 1–13, 2012.

[21] J. Hellerstein, Y. Diao, S. Parekh, and D. Tilbury, *Feedback Control of Computing Systems*. Wiley Online Library, 2004.

[22] W. Lloyd, S. Pallickara, O. David, J. Lyon, M. Arabi, and K. Rojas, "Performance implications of multi-tier application deployments on infrastructure-as-a-service clouds: Towards performance modeling," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1254–1264, 2012.

[23] A. Chandra, W. Gong, and P. Shenoy, "Dynamic Resource Allocation for Shared Data Centers Using Online Measurements," in *Proc. 11th Int. Conf. on Quality of Service*, pp. 381–398, Springer-Verlag, 2003.

[24] P. Xiong, Z. Wang, G. Jung, and C. Pu, "Study on Performance Management and Application Behavior in Virtualized Environment," in *IEEE Netw. Operations and Management Symp.*, pp. 841–844, 2010.

[25] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An Analytical Model for Multi-Tier Internet Services and Its Applications," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 291–302, 2005.

[26] H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, and L. Zhou, "Distributed Systems Meet Economics: Pricing in the Cloud," in *Proc. 2nd USENIX conf. on Hot topics in cloud computing*, pp. 1–6, 2010.

[27] S. Ibrahim, B. He, and H. Jin, "Towards Pay-As-You-Consume Cloud Computing," in *Proc. IEEE Int. Conf. on Services Computing*, pp. 370–377, 2011.

[28] P. Padala, K. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem, "Adaptive Control of Virtualized Resources in Utility Computing Environments," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3, pp. 289–302, 2007.

[29] L. Wang, J. Xu, M. Zhao, and J. Fortes, "Adaptive Virtual Resource Management with Fuzzy Model Predictive Control," in *Proc. 8th ACM Int. Conf. on Autonomic Computing*, pp. 191–192, 2011.

[30] C. Lu, Y. Lu, T. Abdelzaher, J. Stankovic, and S. Son, "Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers," *IEEE Trans. Parallel Distrib. Syst.*, pp. 1014–1027, 2006.

[31] J. Martínez and E. Ipek, "Dynamic Multicore Resource Management: A Machine Learning Approach," *IEEE Micro*, vol. 29, no. 5, pp. 8–17, 2009.

[32] H. Nakada, T. Hirofuchi, H. Ogawa, and S. Itoh, "Toward Virtual Machine Packing Optimization Based on Genetic Algorithm," *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, pp. 651–654, 2009.

[33] S. Kundu, R. Rangaswami, K. Dutta, and M. Zhao, "Application Performance Modeling in a Virtualized Environment," in *IEEE 16th Int. Symp. on High Performance Computer Architecture (HPCA)*, pp. 1–10, 2010.

[34] G. Tesauro, N. Jong, R. Das, and M. Bennani, "A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation," in *IEEE Int. Conf. on Autonomic Computing*, pp. 65–73, 2006.

[35] P. Bodík, R. Griffith, C. Sutton, A. Fox, M. Jordan, and D. Patterson, "Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters," in *Proc. Conf. on Hot Topics in Cloud Computing*, pp. 12–12, USENIX Association, 2009.

[36] C.-Z. Xu, J. Rao, and X. Bu, "Url: A unified reinforcement learning approach for autonomic cloud management," *J. Parallel and Distributed Computing*, vol. 72, no. 2, pp. 95–105, 2012.

[37] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *Int. Conf. on Network and Service Management (CNSM)*, pp. 9–16, IEEE, 2010.

[38] Y. Diao, J. Hellerstein, and S. Parekh, "Optimizing Quality of Service Using Fuzzy Control," *Management Technologies for E-Commerce and E-Business Applications*, pp. 42–53, 2002.

[39] P. Lama and X. Zhou, "Autonomic Provisioning with Self-Adaptive Neural Fuzzy Control for End-to-End Delay Guarantee," in *IEEE Int. Symp. on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 151–160, 2010.

[40] J. Wei and C. Xu, "eQoS: Provisioning of Client-Perceived End-to-End QoS Guarantees in Web Servers," *IEEE Trans. Comput.*, vol. 55, no. 12, pp. 1543–1556, 2006.

[41] J. Rao, Y. Wei, J. Gong, and C. Xu, "Qos guarantees and service differentiation for dynamic cloud applications," *IEEE Trans. Netw. and Service Management*, vol. 10, no. 1, pp. 43–55, 2013.

[42] M. Maurer, I. Brandic, and R. Sakellariou, "Self-adaptive and resource-efficient sla enactment for cloud computing infrastructures," in *IEEE 5th Int. Conf. on Cloud Computing (CLOUD)*, pp. 368–375, 2012.

[43] P. Lama and X. Zhou, "Efficient server provisioning with control for end-to-end response time guarantee on multitier clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 78–86, 2012.

[44] J. Cao, W. Zhang, and W. Tan, "Dynamic control of data streaming and processing in a virtualized environment," *IEEE Trans. Automation Science and Engineering*, vol. 9, no. 2, pp. 365–376, 2012.

[45] L. Wang, J. Xu, M. Zhao, Y. Tu, and J. A. Fortes, "Fuzzy modeling based resource management for virtualized database systems," in *IEEE 19th Int. Symp. on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 32–42, 2011.

[46] B. Kuo and M. Golnaraghi, *Automatic Control Systems*, vol. 1. Wiley Hoboken, NJ, 2003.

[47] M. Maggio, H. Hoffmann, M. Santambrogio, A. Agarwal, and A. Leva, "Decision Making in Autonomic Computing Systems: Comparison of Approaches and Techniques," in *Proc. 8th ACM Int. Conf. on Autonomic Computing*, pp. 201–204, 2011.

[48] W. Song, Z. Xiao, Q. Chen, and H. Luo, "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Trans. Comput.*, vol. 99, 2013.

[49] P. Lama and X. Zhou, "Autonomic provisioning with self-adaptive neural fuzzy control for percentile-based delay guarantee," *ACM Trans. Autonomous and Adaptive Systems (TAAS)*, vol. 8, no. 2, pp. 1–31, 2013.

[50] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, and K. Shin, "What Does Control Theory Bring to Systems Research?," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 1, pp. 62–69, 2009.

[51] Y. Lu, T. Abdelzaher, C. Lu, L. Sha, and X. Liu, "Feedback Control with Queueing-Theoretic Prediction for Relative Delay Guarantees in Web Servers," in *The 9th IEEE Real-Time and Embedded Technology and Applications Symp.*, pp. 208–217, 2003.

[52] T. Abdelzaher, Y. Lu, R. Zhang, and D. Henriksson, "Practical Application of Control Theory to Web Services," in *Proc. of American Control Conf.*, vol. 3, pp. 1992–1997, IEEE, 2004.

[53] K. Sim, "A Survey of Bargaining Models for Grid Resource Allocation," *ACM SIGecom Exchanges*, vol. 5, no. 5, pp. 22–32, 2006.

[54] K. Krauter, R. Buyya, and M. Maheswaran, "A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing," *Software: Practice and Experience*, vol. 32, no. 2, pp. 135–164, 2002.

[55] R. Sharma, V. Soni, M. Mishra, and P. Bhuyan, "A Survey of Job Scheduling and Resource Management in Grid Computing," *World Academy of Science, Engineering and Technology*, vol. 64, pp. 461–466, 2010.

[56] A. C. Zhou and B. He, "Transformation-based monetary cost optimizations for workflows in the cloud," *IEEE Trans. Cloud Computing*, vol. pp, no. 99, pp. 1–1, 2014.

[57] S. Adler, "The Slashdot Effect: An Analysis of Three Internet Publications," *Linux Gazette*, vol. 38, p. 2, 1999.

[58] P. Xiong, Z. Wang, S. Malkowski, Q. Wang, D. Jayasinghe, and C. Pu, "Economical and robust provisioning of n-tier cloud workloads: A multi-level control approach," in *31st Int. Conf. on Distributed Computing Systems*, pp. 571–580, IEEE, 2011.

[59] Rackspace, "http://www.rackspace.com/," 2012.

[60] G. Tesauro, "Online Resource Allocation Using Decompositional Reinforcement Learning," in *Proc. 20th National Conf. on Artificial Intelligence*, vol. 20, pp. 886–891, AAAI Press, 2005.

[61] X. Bu, J. Rao, and C. Xu, "A Reinforcement Learning Approach to Online Web Systems Auto-Configuration," in *29th IEEE Int. Conf. on Distributed Computing Systems*, pp. 2–11, 2009.

[62] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[63] S. S. Keerthi, O. Chapelle, and D. DeCoste, "Building support vector machines with reduced classifier complexity," *The J. of Machine Learning Research*, vol. 7, pp. 1493–1515, 2006.

[64] Y.-C. Wu, Y.-S. Lee, and J.-C. Yang, "Robust and efficient multiclass svm models for phrase pattern recognition," *Pattern recognition*, vol. 41, no. 9, pp. 2874–2889, 2008.

[65] M. Hu, Y. Chen, and J.-Y. Kwok, "Building sparse multiple-kernel svm classifiers," *IEEE Trans. Neural Netw.*, vol. 20, no. 5, pp. 827–839, 2009.

[66] J. Wildstrom, P. Stone, E. Witchel, and M. Dahlin, "Machine Learning for On-Line Hardware Reconfiguration," in *Proc. 20th Int. Joint Conf. On Artificial Intelligence*, pp. 1113–1118, 2007.

[67] O. Niehörster, A. Krieger, J. Simon, and A. Brinkmann, "Autonomic Resource Management with Support Vector Machines," in *12th IEEE/ACM Int. Conf. on Grid Computing*, pp. 157–164, 2011.

[68] S. Kundu, R. Rangaswami, A. Gulati, M. Zhao, and K. Dutta, "Modeling virtualized applications using machine learning techniques," *ACM SIGPLAN Notices*, vol. 47, no. 7, pp. 3–14, 2012.

[69] Q. Noorshams, D. Bruhn, S. Kounev, and R. Reussner, "Predictive performance modeling of virtualized storage systems using optimized statistical regression techniques," in *Proc. ACM/SPEC Int. conf. on performance engineering*, pp. 283–294, 2013.

[70] A. A. B. S. A. Ajila, "Cloud client prediction models for cloud resource provisioning in a multitier web application environment," in *IEEE 7th Int. Symp. on Service-Oriented System Engineering*, pp. 156–161, 2013.

[71] R. Hu, J. Jiang, G. Liu, and L. Wang, "Kswsvr: A new load forecasting method for efficient resources provisioning in cloud," in *IEEE Int. Conf. on Services Computing*, pp. 120–127, 2013.

[72] I. Menache, S. Mannor, and N. Shimkin, "Q-cut?dynamic discovery of sub-goals in reinforcement learning," in *Machine Learning*, pp. 295–306, Springer, 2002.

[73] N. Mehta, S. Ray, P. Tadepalli, and T. Dietterich, "Automatic discovery and transfer of maxq hierarchies," in *Proc. 25th Int. conf. on Machine learning*, pp. 648–655, ACM, 2008.

[74] C.-C. Chiu and V.-W. Soo, "Automatic complexity reduction in reinforcement learning," *Computational Intelligence*, vol. 26, no. 1, pp. 1–25, 2010.

[75] K. Passino and S. Yurkovich, *Fuzzy Control*. Menlo Park, CA: Addison Wesley Longman, 1998.

[76] K. Tanaka and H. Wang, *Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach*. Wiley-Interscience, 2001.

[77] X. Liu, L. Sha, Y. Diao, S. Froehlich, J. Hellerstein, and S. Parekh, "Online Response Time Optimization of Apache Web Server," *Quality of ServicełIWQoS*, pp. 461–478, 2003.

[78] P. Lama and X. Zhou, "Efficient Server Provisioning with End-to-End Delay Guarantee on Multi-Tier Clusters," in *17th Int. Workshop on Quality of Service*, pp. 1–9, IEEE, 2009.

[79] Z. Wang, X. Zhu, and S. Singhal, "Utilization and SLO-Based Control for Dynamic Sizing of Resource Partitions," *Ambient Networks*, pp. 133–144, 2005.

[80] X. Liu, X. Zhu, S. Singhal, and M. Arlitt, "Adaptive Entitlement Control of Resource Containers on Shared Servers," in *IEEE Int. Symp. on Integrated Network Management*, pp. 163–176, 2005.

[81] S. Parekh, N. Gandhi, J. Hellerstein, D. Tilbury, T. Jayram, and J. Bigus, "Using Control Theory to Achieve Service Level Objectives in Performance Management," *Real-Time Systems*, vol. 23, no. 1, pp. 127–141, 2002.

[82] N. Gandhi, D. Tilbury, Y. Diao, J. Hellerstein, and S. Parekh, "MIMO Control of an Apache Web Server: Modeling and Controller Design," in *Proc. of American Control Conf.*, vol. 6, pp. 4922–4927, IEEE, 2002.

[83] L. Cherkasova and P. Phaal, "Session-Based Admission Control: A Mechanism for Peak Load Management of Commercial Web Sites," *IEEE Trans. Comput.*, pp. 669–685, 2002.

[84] M. Karlsson, C. Karamanolis, and X. Zhu, "Triage: Performance Isolation and Differentiation for Storage Systems," in *12th IEEE Int. Workshop on Quality of Service*, pp. 67–74, 2004.

[85] A. Robertsson, B. Wittenmark, M. Kihl, and M. Andersson, "Design and Evaluation of Load Control in Web Server Systems," in *Proc. American Control Conf.*, vol. 3, pp. 1980–1985, IEEE, 2004.

[86] L. Sha, X. Liu, Y. Lu, and T. Abdelzaher, "Queueing Model Based Network Server Performance Control," in *IEEE 23rd Real-Time Systems Symp.*, pp. 81–90, 2002.

[87] Y. Diao, J. Hellerstein, S. Parekh, H. Shaikh, M. Surendra, and A. Tantawi, "Modeling Differentiated Services of Multi-Tier Web Applications," in *14th IEEE Int. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 314–326, 2006.

[88] L. Slothouber, "A Model of Web Server Performance," in *Proc. 5th Int. World Wide Web Conf.*, Citeseer, 1996.

[89] M. Reiser and S. Lavenberg, "Mean-Value Analysis of Closed Multichain Queuing Networks," *J. of the ACM (JACM)*, vol. 27, no. 2, pp. 313–322, 1980.

[90] L. Kleinrock, "Queueing systems, volume i: theory," 1975.

[91] Y. Diao, J. Hellerstein, S. Parekh, R. Griffith, G. Kaiser, and D. Phung, "Self-Managing Systems: A Control Theory Foundation," in *12th IEEE Int. Conf. and Workshops on the Engineering of Computer-Based Systems*, pp. 441–448, 2005.

[92] C. Lu, T. Abdelzaber, J. Stankovic, and S. Son, "A Feedback Control Approach for Guaranteeing Relative Delays in Web Servers," in *7th IEEE Real-Time Technology and Applications Symp.*, pp. 51–62, 2001.

[93] T. Abdelzaher, K. Shin, and N. Bhatti, "Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach," *IEEE Trans. Parallel Distrib. Syst.*, pp. 80–96, 2002.

[94] T. Abdelzaher and K. Shin, "Qos Provisioning with $q$Contracts in Web and Multimedia Servers," in *The 20th IEEE Real-Time Systems Symp.*, pp. 44–53, 1999.

[95] L. Ljung, *System Identification: Theory for the User*, vol. 11. Prentice-Hall Englewood Cliffs, NJ, 1987.

[96] "Matlab System Identification Toolbox." http://www.mathworks.com/products/sysid/, 1988.

[97] Online, "Linux VServer." http://linux-vserver.org/, 2011.

[98] X. Liu, X. Zhu, P. Padala, Z. Wang, and S. Singhal, "Optimal Multivariate Control for Differentiated Services on a Shared Hosting Platform," in *46th IEEE Conf. on Decision and Control*, pp. 3792–3799, 2007.

[99] Y. Zhang, A. Bestavros, M. Guirguis, I. Matta, and R. West, "Friendly Virtual Machines: Leveraging a Feedback-Control Model for Application Adaptation," in *Proc. 1st ACM/USENIX Int. Conf. on Virtual Execution Environments*, pp. 2–12, ACM, 2005.

[100] Y. Song, Y. Sun, H. Wang, and X. Song, "An Adaptive Resource Flowing Scheme amongst VMs in a VM-Based Utility Computing," *7th IEEE Int. Conf. on Computer and Information Technology*, pp. 1053–1058, 2007.

[101] Q. Zhu and G. Agrawal, "Resource Provisioning with Budget Constraints for Adaptive Applications in Cloud Environments," in *Proc. 19th ACM Int. Symp. on High Performance Distributed Computing*, pp. 304–307, 2010.

[102] Q. Zhang, L. Cherkasova, and E. Smirni, "A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications," in *4th Int. Conf. on Autonomic Computing*, pp. 27–27, IEEE, 2007.

[103] P. Lama and X. Zhou, "Perfume: power and performance guarantee with fuzzy mimo control in virtualized servers," in *IEEE 19th Int. Workshop on Quality of Service*, pp. 1–9, 2011.

[104] E. Lazowska, J. Zahorjan, G. Graham, and K. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc., 1984.

[105] P. Lama, Y. Guo, and X. Zhou, "Autonomic performance and power control for co-located web applications on virtualized servers," in *IEEE/ACM 21st Int. Symp. on Quality of Service*, pp. 1–10, 2013.

[106] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey, "Early Observations on the Performance of Windows Azure," in *Proc. 19th ACM Int. Symp. on High Performance Distributed Computing*, pp. 367–376, 2010.

[107] Amazon, "Amazon Auto Scaling Service." http://aws.amazon.com/autoscaling/, 2011.

[108] Rightscale, "Rightscale web site." http://www.rightscale.com, 2011.

[109] H. Lim, S. Babu, J. Chase, and S. Parekh, "Automated Control in Cloud Computing: Challenges and Opportunities," in *Proc. 1st Workshop on Automated Control for Datacenters and Clouds*, pp. 13–18, ACM, 2009.

[110] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, "Dynamic Provisioning of Multi-Tier Internet Applications," in *IEEE 2nd Int. Conf. on Autonomic Computing*, pp. 217–228, 2005.

[111] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal cloud resource autoscaling for web applications," in *13th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing*, pp. 58–65, 2013.

[112] S. Zaman and D. Grosu, "Combinatorial auction-based allocation of virtual machine instances in clouds," *J. of Parallel and Distributed Computing*, vol. 73, pp. 495–508, 2012.

[113] H. Lim, S. Babu, and J. Chase, "Automated Control for Elastic Storage," in *Proc. 7th Int. Conf. on Autonomic Computing*, pp. 1–10, ACM, 2010.

[114] J.-J. E. Slotine and W. Li, *Applied nonlinear control*. Prentice Hall, 1991.

[115] A. J. van der Schaft, *L2-gain and passivity techniques in nonlinear control*. Springer, 2000.

[116] M. Mao and M. Humphrey, "Auto-scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows," in *Proc. 2011 Int. Conf. for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, ACM, 2011.

[117] A. Moshchuk and H. J. Wang, "Resource management for web applications in serviceos," tech. rep., Tech. rep., Microsoft Research, 2010.