Feedback Control-Based Self-Adaptive Cloud Resource Provisioning for Multi-tier Web Systems

Yamin Lei*, Jinquan Zhang†, Long Chen*, Zhicheng Cai‡, and Rajkumar Buyya §
*School of Computer Science and Engineering, Southeast University, Nanjing, China
†School of Computer Science and Technology, Guangdong University of Technology, Guangzhou, China
‡School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
§ The Cloud Computing and Distributed Systems (CLOUDS) Laboratory,
School of Computing and Information Systems, The University of Melbourne, Australia
Corresponding Author: Yamin Lei (leilei@seu.edu.cn)

***Abstract*—In cloud environments, diverse services are provided by multi-tier Web systems. A large number of heterogeneous user requests arrive stochastically to access services. Each request triggers a sequence of internal subrequests that are executed following a predefined order. Incoming subrequests at each tier can be processed by a set of homogeneous virtual machines (VMs). Different alternatives for VM allocation result in distinct rental costs. However, stochastic request arrivals, heterogeneous service requirements, and the complex system topology make it challenging to optimally provision VMs for each tier. In this paper, we consider the cloud VM provisioning problem for a multi-tier Web system to minimize the total VM rental cost across all tiers while satisfying the QoS (Quality of Service) stability requirement at each tier. A time-varying linear state-space model is constructed to characterize request dynamics and inter-tier invocation dependencies. Based on this model, we propose a self-adaptive feedback control method (SFC) to accurately determine the optimal number of VMs for each tier. A SFC-based integrated VM provisioning framework is developed to automatically regulate VM allocations across all tiers based on real-time system performance and resource states under dynamic request fluctuations. SFC is evaluated against existing resource provisioning methods on a simulated Web system. Experimental results show that SFC can simultaneously guarantee the expected QoS for all tiers, and outperforms other methods in reducing the mean QoS deviation ratio by more than 29% with only a moderate increase in total VM rental cost.**

***Index Terms*—VM provisioning, QoS management, Feedback control, State-space model, Multi-tier Web systems**

I. Introduction

Recently, more and more services with various QoS (Quality of Service) guarantees have been deployed on multi-tier Web systems. Each tier encapsulates an independent fine-grained atomic service following the SRP (Single Responsibility Principle) [1]. Tiers are constrained by nonlinear precedence relationships [2], rather than linear ones in traditional three-tier Web systems (i.e., presentation tier, application tier, and database tier) [3]. A large number of heterogeneous user requests stochastically arrive at the multi-tier Web system with different service requirements. Each request accesses a specific service by iteratively calling a sequence of tiers according to a predefined logic order [4]. All requests can be processed by elastically renting VMs (Virtual Machines) from cloud providers. Essentially, a multi-tier Web system bridges the gap between users and the cloud provider. Users are satisfied after the QoS requirements are met. In this paper, MRT (Mean Response Time) is used to measure QoS. MRT is considered stable only if it remains close to a desired reference time. The objective is to minimize the total VM rental cost across all tiers while ensuring the expected MRT at each tier.

We propose a self-adaptive VM provisioning solution for a multi-tier Web system under tier-specific MRT stability constraints. The atomic services in the system are invoked sequentially and asynchronously among which data transfer delays are not considered. Each one-to-one invocation generates an internal subrequest issued by the corresponding upstream tier. Subrequests at each tier are processed by a set of homogeneous VMs, whereas heterogeneous VMs can be configured for distinct tiers.

Stochastic request arrivals, heterogeneous service requirements, and the complex topology of the multi-tier Web system make it difficult to optimize VM provisioning. The main challenges in the problem under study are: (i) Stochastic arrivals of requests lead to dynamic fluctuations in MRTs. The MRT at each tier greatly depends on the number of allocated VMs. Since the expected MRTs differ across tiers, it is challenging to determine the VM allocations that can simultaneously satisfy MRT stability constraints across all tiers. (ii) Requests are processed by multiple tiers along different service paths. The uncertainty in arrival times and types of requests poses difficulties in characterizing subrequest dynamics at each tier, which is a prerequisite for VM provisioning. (iii) Subrequests at each tier are processed sequentially on each VM but in parallel across multiple VMs. As a result, more VMs usually incur a higher VM rental cost without ensuring a lower MRT. This trade-off makes it challenging to determine the cost-efficient VM allocation.

In this paper, we develop a self-adaptive VM provisioning framework for stochastically arriving user requests in a multi-tier Web system. The main contributions are:

- A time-varying linear MIMO (Multiple-Input Multiple-Output) state-space model is constructed to characterize request dynamics and inter-tier invocation dependencies.
- A state-space model-driven self-adaptive feedback control method (SFC) is proposed to accurately determine the

optimal number of VMs for each tier.
- An SFC-based integrated provisioning framework is designed to monitor system performance and resource states in real time and automatically regulate VM allocations across all tiers to ensure MRT stability.

The rest of the paper is organized as follows. Section II reviews related work. The system architecture and problem formulation are detailed in Section III. Section IV elaborates the proposed SFC. Experimental results are analyzed in Section V, followed by conclusions and future research in Section VI.

II. Related Work

Numerous cloud resource provisioning approaches have been proposed for multi-tier Web systems with linear or nonlinear typologies. They capture inter-tier invocations by using various techniques, such as service dependency graphs (SDGs) [13], queuing networks [14]–[22], analytical models [23]–[28], reinforcement learning (RL) [29]–[33], and MIMO statistical models [20], [34]–[36]. Based on these models, optimization and control methods are usually applied to determine the optimal resource provisioning decisions.

Since the traffic flows naturally reflect inter-tier invocation relationships, an SDG can be constructed from historical request and subrequest observations. Based on the built SDG, the resource allocation that satisfies the predefined QoS constraints can be determined by using the Bayesian optimization [13] or trust-region algorithm [30]. However, these algorithms may converge to local optima and fail to find a global optimal solution even with substantial iterations. Both queuing networks and analytical models provide an alternative way. Layered queuing networks [37], [38] that are built from transaction activities in the SDG are typically combined with state space model-based controllers [37] or genetic algorithms [38] for resource allocation. Open Jackson queuing networks model the subrequest dynamics at each tier as a queuing model (e.g., G/G/1 model [14] and M/M/N model [16], [17]), while the subrequest routing across tiers is characterized as a Markov process. Extended Jackson queuing networks incorporate proportional controllers [20], request drop probabilities [21], or loop execution probabilities [19] into queuing networks. A "what-if-analysis" based exhaustive search mechanism is then applied to obtain the optimal number of resources at each tier, which hypothetically scales resources one by one until the QoS constraint is met [14], [16], [17], [20]. Compared with Jackson queuing networks, analytical models enable greater flexibility through diverse metrics, such as subrequest access ratios [23], delay proportions [27], [28], Markovian steady-state probabilities [26], and MRT-based performance promises [25]. However, the effectiveness of all these models depends on the underlying queuing model, i.e., the provisioning operations become unreliable if the queuing model deviates from actual subrequest processing behavior.

Both RL [29]–[33] and MIMO statistical techniques [20], [34]–[36] that require no prior knowledge can avoid the above limitations. They integrate the observation data on request amount, resource configuration, and system performance to model inter-tier invocations and make resource provisioning decisions for all tiers. However, RL suffers from the curse of dimensionality, i.e., the exploration time of the state-action space increases exponentially with system complexity and request diversity. MIMO models rely on system identification techniques rather than physical equation derivation, whose training time is much shorter than that of RL. Feedback controllers are usually coupled with MIMO models to optimize resource allocation and stabilize system performance. Therefore, MIMO models are well suited to multi-tier Web systems with complex typologies. However, existing MIMO models are mainly deterministic and linear. They perform well only under static or slightly changing workloads, but fail to capture the full fluctuation patterns of highly dynamic requests considered in this paper. To the best of our knowledge, the resource provisioning problem has never been studied for stochastically arriving heterogeneous requests in multi-tier Web systems with complex service processes and stable MRT requirements.

III. System Architecture and Problem Description

A. System Architecture

For the considered problem, an integrated VM provisioning framework is developed for a multi-tier Web system as depicted in Fig. 1. Loosely coupled multiple tiers in the system communicate with each other through lightweight RESTful APIs (Application Programming Interfaces). A directed invocation link can be created between two different tiers based on system requirements. Usually more than one invocation link composes a service path without loops, i.e., neither self-calls of a tier nor cyclic-calls between tiers are allowed. All invocation links form a directed acyclic graph.

Users stochastically generate multiple types of independent requests to access the multi-tier Web system. Requests are initially queued in the front-end API Gateway and are sequentially processed in a first-come, first-served (FCFS) manner without rejection. Each request is processed in a serial and distributed manner, i.e., the whole process contains multiple stages and each stage operates at a specific tier. Specifically, the API Gateway analyzes the request type to determine the service path. All tiers in the chain are successively accessed. Once the atomic service at a tier is completed, all related data is encapsulated into a single subrequest. This subrequest is sent to the immediate successor tier to asynchronously call the corresponding atomic service. This process is repeated until the complete service requirement of the original request is met. Different types of requests have distinct service requirements on different service paths.

Take a geo-intelligent recommendation scenario as an example. The API Gateway L_0 receives a request q and identifies the service path as (L_1, L_2, L_3). L_1, L_2, and L_3 denote Search Engine, Business Recommendation, and the Location Mapping Tiers, respectively. Initially, L_0 augments q with the service path information to form a subrequest q_1 and forwards it to L_1.

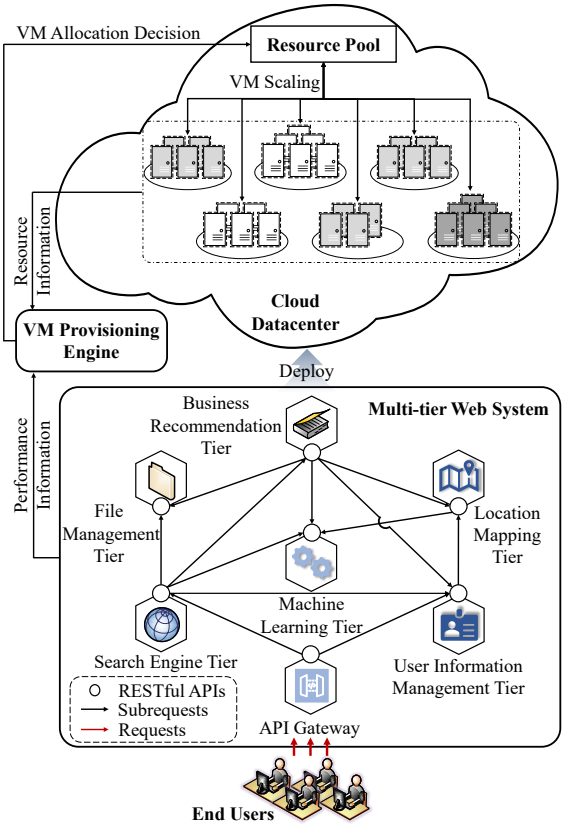


Fig. 1. VM provisioning framework for a representative multi-tier Web system in cloud environment.

L_1 extracts keywords (e.g., city and hotel) from q_1 and encapsulates all retrieved hotel information into a new subrequest q_2. q_2 is then sent to L_2 for personalized recommendation based on user preferences. L_2 integrates all recommended hotel information to generate another subrequest q_3 and forwards it to L_3. L_3 provides surrounding details and traffic guidance for each recommended hotel. The processing of q concludes after all three tiers in the service path successively complete their respective local atomic services.

Each tier is deployed on a set of homogeneous on-demand VMs from the cloud provider. VMs across different tiers may be heterogeneous. Following the FCFS rule, all incoming subrequests at a tier are scheduled to rented VMs in a stateless mode under load balancing. Subrequests on each VM are processed in sequence based on the FCFS rule, while multiple subrequests across different VMs can be processed in parallel.

The VM Provisioning Engine regularly monitors the performance and resource states in the multi-tier Web system. Based on this runtime information, it automatically generates an optimal VM allocation decision that specifies the number of VMs for each tier.

B. Problem Description

Assume that the multi-tier Web system comprises the API Gateway L_0 and $\mathbb{L}$ tiers. L_0 is only responsible for request parsing and routing whose VM scaling is not considered. Let $r_i(k)$ be the real MRT of all subrequests accessing a tier $L_i, i \in I$, where $I = \{1, 2, \cdots, \mathbb{L}\}$ is the set of tier identifiers. The MRT deviation ratio $\mathcal{D}_i(k)$ in each interval measures the extent to which $r_i(k)$ deviates from the desired time reference $\mathbb{R}_i^r$. The MRT stability constraint specifies that the mean deviation ratio $\bar{\mathcal{D}}_i$ across the entire system hosting period (consisting of $\mathbb{N}$ identical intervals) cannot exceed the predefined threshold $\mathbb{D}^{sla}$. Mathematically,

$$\mathcal{D}_i(k) = \frac{|r_i(k) - \mathbb{R}_i^r|}{\mathbb{R}_i^r} \times 100\%, \tag{1}$$

$$\bar{\mathcal{D}}_i = \frac{\sum_{k=1}^{\mathbb{N}} \mathcal{D}_i(k)}{\mathbb{N}} \leq \mathbb{D}^{sla}. \tag{2}$$

Since there is no VM sharing across tiers, the total VM rental cost C equals the sum of the VM rental costs across all tiers. Let $\mathcal{C}_i$ be the rental cost at the tier L_i, the considered cost minimization problem can be modeled as follows:

$$\begin{aligned} \min \quad & C = \sum_{i=1}^{\mathbb{L}} \mathcal{C}_i = \sum_{i=1}^{\mathbb{L}} \sum_{k=1}^{\mathbb{N}} \mathbb{P}_i \times \mathbb{T} \times n_i(k) \\ s.t. \quad & \bar{\mathcal{D}}_i \leq \mathbb{D}^{sla}, \forall i \in I, \end{aligned} \tag{3}$$

where $\mathbb{T}$ represents the length of each interval. $n_i(k)$ denotes the number of VMs provisioned at L_i in the k^{th} interval and $\mathbb{P}_i$ denotes the unit price per minute per VM.

IV. Proposed Algorithm

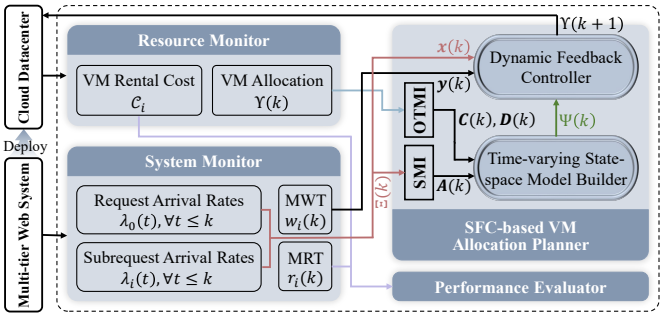
A multi-tier Web system typically exhibits nonlinearity due to the stochastic arrival of heterogeneous requests and the complex topology structure. It is challenging to design a fully nonlinear feedback control method to guarantee the MRT stability requirements for all tiers. In this paper, we propose a time-varying linear state-space model-based self-adaptive feedback control method (SFC in short). SFC operates at the core of the VM Provisioning Engine which consists of four modules as shown in Fig. 2. At the end of each interval, the SFC-based VM Allocation Planner collects all observation data from the Resource and the System Monitors, and applies the SFC method to automatically generate an optimal VM allocation decision $\Upsilon(k+1) = \{n_1(k+1), \cdots, n_{\mathbb{L}}(k+1)\}$ for the next interval. $n_i(k+1), i \in I$ specifies the expected number of VMs allocated to each tier L_i. The Performance Evaluator assesses the performance of SFC based on MRTs and rental costs across all tiers.

A. Time-varying State-space Model Builder

SFC builds a time-varying discrete linear MIMO state-space model $\Psi(k)$ as follows:

$$\Psi(k): \begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{A}(k)\boldsymbol{x}(k) \\ \boldsymbol{y}(k) = \boldsymbol{C}(k)\boldsymbol{x}(k) + \boldsymbol{D}(k)\boldsymbol{u}(k), \end{cases} \tag{4}$$

where the state vector consists of the request arrival rate $\lambda_0(k)$ at the API Gateway L_0 and subrequest arrival rates at all tiers, i.e., $\boldsymbol{x}(k) = (\lambda_0(k), \lambda_1(k), \cdots, \lambda_{\mathbb{L}}(k))^\top$. The control input vector consists of the total VM processing rates at all tiers, i.e., $\boldsymbol{u}(k) = (\mu_0(k), \mu_1(k), \cdots, \mu_{\mathbb{L}}(k))^\top$. Since MWTs better reflect the suberquest blocking on provisioned VMs



Note: The indices $i \in I$ and $1 \le k \le \mathbb{N}$ apply to all relevant variables in this figure.

Fig. 2. Architecture of the SFC-based VM provisioning engine.

than MRTs, the control output vector is built from MWTs, i.e., $\boldsymbol{y}(k) = (w_0(k), w_1(k), \cdots, w_{\mathbb{L}}(k))^\top$. $\mu_0(k) = 0$ and $w_0(k) = 0$ always hold since the VM provisioning is not considered at L_0.

1) State Matrix Identification: The time-varying state matrix $\boldsymbol{A}(k)$ is built to capture the dynamic fluctuations in requests and subrequests. The request dynamics can be described as a difference equation

$$\lambda_0(k+1) = \xi_0 \lambda_0(k), 0 < \xi_0 < 1, \tag{5}$$

where the coefficient ξ_0 indicates the change rate of request arrival rates in two adjacent intervals. At each tier L_i $(i \in I)$, incoming subrequests are generated by all its upstream tiers, which essentially originate from L_0. Mathematically,

$$\lambda_i(k) = \varsigma_i^k \lambda_0(k), 0 \le \varsigma_i^k \le 1, i \in I, \tag{6}$$

where ς_i^k indicates the probability that requests access L_i along all feasible service paths in the k^{th} interval. The subrequest dynamics can be described by combing Eqs. (5) and (6):

$$\lambda_i(k+1) = \varsigma_i^{k+1} \xi_0 \lambda_0(k), 0 \le \varsigma_i^{k+1} \le 1, 0 < \xi_0 < 1, i \in I. \tag{7}$$

Based on Eqs. (4), (5), and (7), $\boldsymbol{A}(k)$ can be modeled as a $(\mathbb{L}+1) \times (\mathbb{L}+1)$ matrix. $a_{00}(k) = \xi_0$ is determined offline from historical request arrival rates by applying a least squares regression-based system identification technique [39], [40]. $a_{i0}(k) = \varsigma_i^{k+1} \xi_0$ $(i \in I)$ is updated online by using a sliding window of size τ (e.g., $\tau = 6$). When $\tau \le k \le \mathbb{N}$, all historical request arrival rates in the window form a time series $\mathcal{HA}_0(k) = [\lambda_0(k-\tau+1), \cdots, \lambda_0(k-1), \lambda_0(k)]$. The corresponding subrequest arrival rates at L_i form a time series $\mathcal{HA}_i(k)$. $a_{i0}(k)$ is determined by directly calling the built-in least squares regression function $leastsq(\mathcal{HA}_i(k), \mathcal{HA}_0(k))$. When $1 \le k < \tau$, there is insufficient historical data and an initial constant matrix $\boldsymbol{A}_I$ is introduced. Algorithm 1 describes the complete process.

2) Output and Direct Transfer Matrix Identification: Subrequests at each tier fluctuate dynamically, which implies that VMs need to be provisioned elastically. The nonlinear relationship among the subrequest arrival rate $\lambda_i(k), i \in I$, the total VM processing rate $\mu_i(k)$, and the MWT $w_i(k)$ [41] is commonly approximated as a linear model $w_i(k) = c_i \lambda_i(k) + d_i \mu_i(k)$ [20]. However, such deterministic linearization always introduces unavoidable deviations between calculated and observed MWTs. Fortunately, the nonlinear relationship usually exhibits a piecewise linearity, which can be modeled as a piecewise linear function:

$$w_i(k) = \begin{cases} c_{ii}^{v_i^l} \lambda_i(k) + d_{ii}^{v_i^l} \mu_i(k) & n_i(k) = v_i^l \\ \vdots & \vdots \\ c_{ii}^{v_i} \lambda_i(k) + d_{ii}^{v_i} \mu_i(k) & n_i(k) = v_i \\ \vdots & \vdots \\ c_{ii}^{v_i^u} \lambda_i(k) + d_{ii}^{v_i^u} \mu_i(k) & n_i(k) = v_i^u \end{cases}. \tag{8}$$

Values $v_i^l, \cdots, v_i^u, \forall i \in I$ are extracted from historical data. Each VM configuration $n_i(k)$ corresponds to a locally linear segment $\mathcal{S}_i^{v_i}, v_i \in \{v_i^l, \cdots, v_i^u\}$, whose parameters $c_{ii}^{v_i}$ and $d_{ii}^{v_i}$ are determined offline by using the least squares regression technique [39], [40]. All segments form a set $\mathcal{PS}_i = \{(v_i, (c_{ii}^{v_i}, d_{ii}^{v_i})) \mid v_i \in \{v_i^l, \cdots, v_i^u\}\}$.

At the end of each interval, L_i automatically switches across segments. In other words, L_i can always select the most appropriate values for $c_{ii}(k)$ and $d_{ii}(k)$ according to the current number of VMs. Considering that no VMs are shared across tiers, both output matrix $\boldsymbol{C}(k)$ and direct transfer matrix $\boldsymbol{D}(k)$ can be built as $(\mathbb{L}+1)\times(\mathbb{L}+1)$ diagonal matrices. $c_{ii}(k)$ and $d_{ii}(k)$ are the $(i+1)^{th}$ diagonal elements in $\boldsymbol{C}(k)$ and $\boldsymbol{D}(k)$, respectively. To ensure the controllability of all unstable poles in $\Psi(k)$, $c_{00}(k) = 0$ and $d_{00}(k) = 1$ are always set. The formal process is detailed in Algorithm 2.

Algorithm 1 State Matrix Identification Method (SMI)

Input: $\Xi(k) = \{\mathcal{HA}_0(k), \mathcal{HA}_1(k), \cdots, \mathcal{HA}_{\mathbb{L}}(k)\}$
Output: $\boldsymbol{A}(k)$

```
Initialize A(k) ← 0_{(L+1)×(L+1)};
for each interval k, 1 ≤ k ≤ N do
    if 1 ≤ k < τ then
        A(k) ← A_I;
    else
        a_00(k) ← ξ_0;
        for each tier L_i, i ∈ I do
            a_i0(k) ← leastsq(HA_i(k), HA_0(k));
        end for
    end if
end for
return A(k)
```

B. Dynamic Feedback Controller

Dynamic feedback controllers have been widely used for MIMO systems to achieve performance stability. We derive the control law by considering the dual impacts of subrequest fluctuations and integrated control error (ICE) on VM allocations. Mathematically,

$$\boldsymbol{u}(k) = -\begin{bmatrix} \boldsymbol{K}_P(k) & \boldsymbol{K}_I(k) \end{bmatrix} \times \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{e_I}(k) \end{bmatrix}. \tag{9}$$

Algorithm 2 Output and Direct Transfer Matrix Identification Method (OTMI)

Input: $\Upsilon(k) = \{n_1(k), \cdots, n_{\mathbb{L}}(k)\}$
Output: $\boldsymbol{C}(k)$, $\boldsymbol{D}(k)$

Initialize $\boldsymbol{C}(k) \leftarrow \mathbf{0}_{(\mathbb{L}+1)\times(\mathbb{L}+1)}$, $\boldsymbol{D}(k) \leftarrow \mathbf{0}_{(\mathbb{L}+1)\times(\mathbb{L}+1)}$;
for each interval $k, 1 \leq k \leq \mathbb{N}$ **do**
  $c_{00}(k) \leftarrow 0$, $d_{00}(k) \leftarrow 1$;
  for each tier L_i, $i \in I$ **do**
    Search $c_{ii}^{n_i(k)}$, $d_{ii}^{n_i(k)}$ from $\mathcal{PS}_i$ based on $n_i(k)$;
    $c_{ii}(k) \leftarrow c_{ii}^{n_i(k)}, d_{ii}(k) \leftarrow d_{ii}^{n_i(k)}$;
  end for
end for
return $\boldsymbol{C}(k)$, $\boldsymbol{D}(k)$

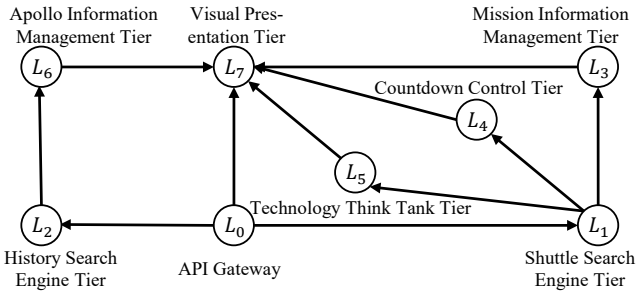


Fig. 3. Architecture of the NASA multi-tier Web system.

$\boldsymbol{K}_P(k)$ and $\boldsymbol{K}_I(k)$ are control gains in the k^{th} interval. LQR (Linear Quadratic Regulator) [20], [40] can be applied to determine their values by minimizing the following objective function:

$$J = \frac{1}{2}\sum_{k=0}^{\infty}[\boldsymbol{s}^\top(k)\boldsymbol{Q}(k)\boldsymbol{s}(k) + \boldsymbol{u}^\top(k)\boldsymbol{R}(k)\boldsymbol{u}(k)], \quad (10)$$

where $\boldsymbol{s}(k) = [\boldsymbol{x}(k), \boldsymbol{e_I}(k)]^\top$. Two weighting matrices $\boldsymbol{Q}(k)$ and $\boldsymbol{R}(k)$ are introduced to balance control efforts and control errors. It is required that $\boldsymbol{Q}(k)$ be positive semi-definite and $\boldsymbol{R}(k)$ be positive definite. ICE accumulates the control errors of all previous intervals, i.e.,

$$\boldsymbol{e_I}(k) = \boldsymbol{e_I}(k-1) + \boldsymbol{e}(k-1). \quad (11)$$

Let $\boldsymbol{r} = (\mathbb{W}_0^r, \mathbb{W}_1^r, \cdots, \mathbb{W}_{\mathbb{L}}^r)^\top$ be the vector of desired reference MWTs. SFC calculates $\boldsymbol{e}(k-1) = (e_0(k-1), e_1(k-1), \cdots, e_{\mathbb{L}}(k-1))^\top$ as the difference between $\boldsymbol{r}$ and the observed MWT $\boldsymbol{y}(k)$, i.e., $\boldsymbol{e}(k-1) = \boldsymbol{r} - \boldsymbol{y}(k-1)$.

SFC couples this dynamic feedback controller with the linear state-space model $\Psi(k)$ to build a time-varying closed-loop servo system (all components in Fig. 2 form a closed loop). At the end of each interval, $\boldsymbol{u}(k+1)$ for the next interval can be determined based on Eqs. (4), (9), and (11). The complete process is described in Algorithm 3. For each tier $L_i, i \in I$, $n_i(k+1)$ is calculated as $\lceil \mu_i(k+1)/\bar{\mu}_i \rceil$ since homogeneous VMs are provisioned for each tier, where $\bar{\mu}_i$ denotes the mean processing rate of each VM at L_i.

Algorithm 3 Time-varying Linear State-space Model-based Self-adaptive Feedback Control Method (SFC)

Input: $\Xi(k)$, $\Upsilon(k)$, $\boldsymbol{x}(k)$, $\boldsymbol{y}(k)$, $\boldsymbol{r}$
Output: $\Upsilon(k+1)$

Initialize $\Upsilon(k+1) \leftarrow \varnothing$
for each interval $k, 1 \leq k \leq \mathbb{N}$ **do**
  Call SMI($\Xi(k)$) to obtain $\boldsymbol{A}(k)$; /*Algorithm 1*/
  Call OTMI($\Upsilon(k)$) to obtain $\boldsymbol{C}(k)$ and $\boldsymbol{D}(k)$; /*Algorithm 2*/
  Obtain $\boldsymbol{K}_P(k+1)$ and $\boldsymbol{K}_I(k+1)$ by using LQR;
  $\boldsymbol{x}(k+1) = \boldsymbol{A}(k)\boldsymbol{x}(k)$;
  $\boldsymbol{e_I}(k+1) = \boldsymbol{e_I}(k) + \boldsymbol{r} - \boldsymbol{y}(k)$;
  $\boldsymbol{u}(k+1) = -\boldsymbol{K}_P(k+1)\boldsymbol{x}(k+1) - \boldsymbol{K}_I(k+1)\boldsymbol{e_I}(k+1)$;
  for each tier L_i, $i \in I$ **do**
    $n_i(k+1) \leftarrow \lceil \mu_i(k+1)/\bar{\mu}_i \rceil$;
    $\Upsilon(k+1) \leftarrow \Upsilon(k+1) \cup n_i(k+1)$;
  end for
end for
return $\Upsilon(k+1)$

V. Performance Evaluation

Our proposed SFC is compared with three existing provisioning methods designed for multi-tier Web systems. All comparison experiments are conducted on a simulated test-bed, which is established based on the widely used CloudSim platform [42]. This simulated test-bed environment implements the complete VM provisioning framework in Fig. 1.

Various Amazon EC2 on-demand VMs with different configurations and prices are simulated in the Resource Pool of the cloud provider. For simplicity, we provision VMs of type c4.2xL charged at $\$0.007$ per minute for all tiers. However, the mean processing rate of each VM varies across tiers due to different mean sizes of subrequests. This implies that VMs across different tiers remain heterogeneous.

The NASA multi-tier Web system is reconstructed based on publicly available NASA HTTP access traces [43] collected from the Kennedy Space Center Web server. Each URL (Uniform Resource Locator) corresponds to a user request. Each folder name in the URL is regarded as an accessed tier, all of which form the service path of the corresponding request. All service paths collectively constitute the simulated NASA prototype system with seven tiers as shown in Fig. 3. Tiers have different characteristics as summarized in Table I, including the mean subrequest size $\mathbb{S}_i$, the mean VM processing rate $\bar{\mu}_i$, and the reference MRT $\mathbb{R}_i^r$. The reference MWT can be calculated as $\mathbb{W}_i^r = \mathbb{R}_i^r - 1/\bar{\mu}_i$. $\mathbb{D}^{sla} = 5\%$ is set consistently for all tiers. Stochastically arriving user requests in the NASA Web system are also synthesized from realistic HTTP traces. The reactive VM provisioning interval is fixed at $\mathbb{T} = 5$ minutes.

A. Experimental Results

For a fair comparison, all parameters involved in each VM provisioning method are calibrated to their empirically best

TABLE I
TIER CHARACTERISTICS OF THE NASA MULTI-TIER WEB SYSTEM

Tier	$\mathbb{S}_i$(MIPS)	$\bar{\mu}_i$(/s)	$\mathbb{P}_i$($/min)	$\mathbb{R}_i^r$(s)	$\mathbb{W}_i^r$(s)	$\mathbb{D}^{sla}$
L_1 to L_3	500	31	0.007	0.036	0.004	5%
L_4 & L_5	5000	3.1		0.340	0.020	
L_6 & L_7	500	31		0.036	0.004	

values. All methods are initialized with the same VM allocation $\Upsilon(0)$. The Performance Evaluator component assesses the system performance based on multiple metrics as presented in Fig. 2: (i) Mean MRT deviation ratio $\bar{\mathcal{D}}_i$ at each tier $L_i, i \in I$, (ii) VM rental cost $\mathcal{C}_i$ at each tier and the total rental cost $C = \sum_{i=1}^{\mathbb{L}} \mathcal{C}_i$. Results are summarized in Table II and Table III, respectively.

1) Comparisons with Queuing Network-based Provisioning Methods: PPM (Proportional Provisioning Method) [14] models the inter-tier invocation relationships in each interval as a vector $\mathbf{P}(k+1)$ based on the ratios of subrequest arrival rates to the request arrival rate. The estimated subrequest arrival rate $\tilde{\lambda}_i(k+1), i \in I$ is derived from $\mathcal{P}(k+1)$ under an assumption of $\tilde{\lambda}_0(k+1) = \lambda_0(k)$. JPM (Jackson queuing network-based Provisioning Method) [16], [17] builds inter-tier invocations as a Markov transition matrix $\mathbf{P}(k+1)$ based on transition probabilities. Similarly, $\tilde{\lambda}_i(k+1)$ is obtained based on $\mathbf{P}(k+1)$ under the same assumption. Compared to $\mathcal{P}(k+1)$, $\mathbf{P}(k+1)$ can more accurately capture the impact of sudden bottleneck tiers on successor tiers. However, requests considered in this paper fluctuate dynamically without abrupt changes, no bottleneck tiers arise in the NASA multi-tier Web system. This implies that there is no difference between $\mathcal{P}(k+1)$ and $\mathbf{P}(k+1)$, i.e., PPM and JPM generate equal $\tilde{\lambda}_i(k+1)$. Based on $\tilde{\lambda}_i(k+1)$, they adopt different queuing models to determine the number of VMs $n_i(k+1)$. Specifically, JPM adopts a "what-if-analysis" based exhaustive search method, which iteratively increases $n_i(k+1)$ by 1 until the MWT calculated based on the M/M/N model falls below the reference MWT $\mathbb{W}_i^r$. For a fair comparison, we assume that both subrequest inter-arrival times and VM service times follow exponential distributions. In other words, PPM models each VM as an M/M/1 model instead of the original G/G/1 model. The ideal ART $\bar{\lambda}_i$ that a single VM can process at exactly $\mathbb{W}_i^r$ can be derived from the M/M/1 model. $n_i(k+1)$ is then calculated as $\lceil \tilde{\lambda}_i(k+1)/\bar{\lambda}_i \rceil$. Since the fluctuation patterns of subrequests vary across tiers, a single M/M/N model cannot capture this heterogeneity. The actual subrequest inter-arrival times at some tiers deviate from the assumed exponential distribution. As a result, both PPM and JPM misestimate $n_i(k+1)$. $\mathbb{D}^{sla}$ is violated at more than half of the tiers as shown in Table II. Actually, PPM always over-provision VMs, whereas JPM tends to under-provision VMs, as depicted in Figs. 4 and 5. Correspondingly, PPM and JPM generate the highest and lowest VM rental cost, respectively. Compared with queuing models, the state-space model in the SFC method has better advantages in capturing inter-tier invocations and

TABLE II
MEAN MRT DEVIATION RATIOS ACROSS ALL TIERS UNDER ALL PROVISIONING METHODS

Method	$\bar{\mathcal{D}}_1(\%)$	$\bar{\mathcal{D}}_2(\%)$	$\bar{\mathcal{D}}_3(\%)$	$\bar{\mathcal{D}}_4(\%)$	$\bar{\mathcal{D}}_5(\%)$	$\bar{\mathcal{D}}_6(\%)$	$\bar{\mathcal{D}}_7(\%)$
PPM	9.949	9.870	9.484	5.031	**4.834**	9.575	9.699
JPM	**4.268**	7.827	**3.339**	24.642	15.298	5.357	**4.176**
PPN	10.856	17.887	7.991	5.132	**3.458**	7.557	8.762
SFC	**4.074**	**4.588**	**3.695**	**4.958**	**4.463**	**4.986**	**3.734**

Note: All $\bar{\mathcal{D}}_i$ values that satisfy the MRT stability constraint of $\bar{\mathcal{D}}_i < \mathbb{D}^{sla}$ are highlighted in bold.

TABLE III
VM RENTAL COSTS ACROSS ALL TIERS UNDER ALL PROVISIONING METHODS

Method	$\mathcal{C}_1$($)	$\mathcal{C}_2$($)	$\mathcal{C}_3$($)	$\mathcal{C}_4$($)	$\mathcal{C}_5$($)	$\mathcal{C}_6$($)	$\mathcal{C}_7$($)	C($)
PPM	221.07	118.63	140.70	696.64	221.07	99.54	221.07	1718.72
JPM	130.53	65.32	91.80	364.73	112.75	59.09	131.22	955.44
PPN	143.21	61.41	100.52	451.62	147.81	60.94	145.25	1110.76
SFC	135.06	74.76	97.03	405.39	136.13	66.28	137.77	1052.42

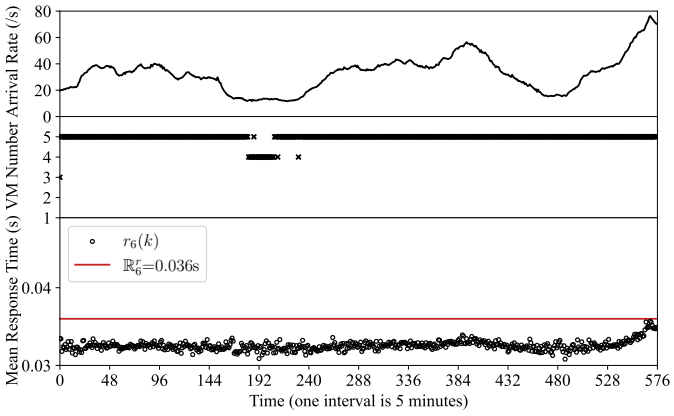


Fig. 4. Subrequest arrival rate, number of VMs, and MRT at L_6 under PPM.

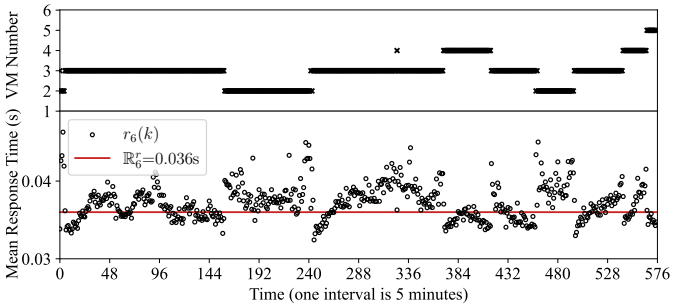


Fig. 5. Number of VMs and MRT at L_6 under JPM.

subrequest dynamics. Therefore, SFC can satisfy the MRT stability requirements for all tiers. Compared with PPM and JPM, it reduces the MRT deviation ratio by 41.73% and 29.07% on average, respectively.

2) Comparisons with Analytical Model-based Provisioning Method: PPN (Performance Promise Negotiation-based provisioning method) [25] builds an analytical model for global VM adjustment across all tiers in the entire Web system.

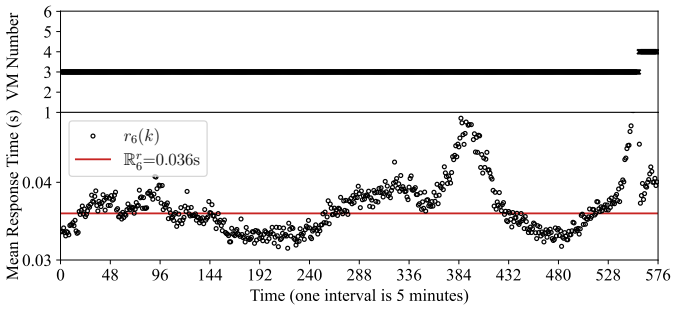


Fig. 6. Number of VMs and MRT at L_6 under PPN. Values of $r_6(k)$ exceeding 0.05s are capped at 0.05s for visualization.

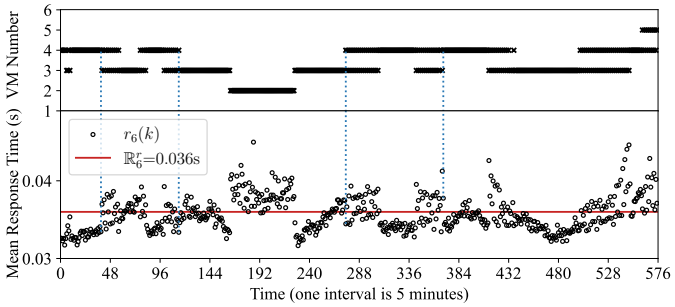


Fig. 7. Number of VMs and MRT at L_6 under SFC.

Specifically, PPN iteratively adds VMs whenever the system-level end-to-end MRT constraint is violated. Each iteration specifies a gain promise $\Delta r_i^g(k)$ for each tier $L_i, i \in I$. $\Delta r_i^g(k)$ is the expected MRT reduction after adding a VM. After all tiers negotiate their $\Delta r_i^g(k)$ values, L_0 adds a VM to the tier with the largest $\Delta r_i^g(k)$. Similarly, PPN iteratively releases VMs based on loss promises $\Delta r_i^l(k), \forall i \in I$, which measure the expected increase in MRT after releasing a VM from L_i.

The promise negotiation process essentially models the inter-tier invocation relationships. Theoretically, such a global method can more flexibly adjust VMs across all tiers. However, $\Delta r_i^g(k)$ and $\Delta r_i^l(k)$ are derived from an M/M/N/PS queuing model, whose inherent limitations as discussed above lead to inaccurate estimations. L_0 cannot identify tiers that truly require VM scaling. The under-provisioning and over-provisioning of VMs may remain undetected for a long period as depicted in Fig. 6. Despite higher rental cost than SFC, $\mathbb{D}^{sla}$ is severely violated at almost all tiers. In contrast, the dynamic feedback controller in the SFC method enables more accurate VM allocations, particularly under subrequest arrivals with higher variability as shown in the highlighted regions in Fig. 7. SFC achieves a 36.60% reduction in the mean MRT deviation ratio compared with PPN.

VI. Conclusions and Future Work

This paper studies VM provisioning for stochastically arriving heterogeneous user requests in multi-tier Web systems. A time-varying linear state-space model-based self-adaptive feedback control method SFC is proposed to simultaneously determine the optimal number of VMs for all tiers. An integrated SFC-based VM provisioning framework is developed to automatically regulate VM allocations across all tiers based on system performance and resource states under dynamic request fluctuations. SFC is evaluated against three existing provisioning algorithms designed for similar problems. Experimental results reveal that the proposed SFC outperforms the others in meeting MRT stability requirements for all tiers at a slightly higher total VM rental cost. This paper focuses on tier-level MRT stability. However, the stable end-to-end MRT is more critical for users in real-world scenarios. Extending VM provisioning to incorporate such system-level requirements is a promising research topic.

References

[1] W. Jin, T. Liu, Y. Cai, R. Kazman, R. Mo, and Q. Zheng, "Service candidate identification from monolithic systems based on execution traces," *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 987–1007, 2021.

[2] S. Luo, H. Xu, C. Lu, K. Ye, G. Xu, L. Zhang, Y. Ding, J. He, and C. Xu, "Characterizing microservice dependency and performance: Alibaba trace analysis," in *Proceedings of the 12th ACM Symposium on Cloud Computing (SoCC)*, 2021, pp. 412–426.

[3] X. Yin, J. Huang, L. Liu, W. He, and L. Cui, "An iterative feedback mechanism for auto-optimizing software resource allocation in multi-tier web systems," in *20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, 2020, pp. 802–809.

[4] S. Luo, H. Xu, K. Ye, G. Xu, L. Zhang, J. He, G. Yang, and C. Xu, "Erms: Efficient resource management for shared microservices with sla guarantees," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2023, pp. 62–77.

[5] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. Katz, "Napsac: Design and implementation of a power-proportional web cluster," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 102–108, 2011.

[6] F. Zhang, X. Tang, X. Li, S. U.Khan, and Z. Li, "Quantifying cloud elasticity with container-based autoscaling," *Future Generation Computer Systems*, vol. 98, pp. 672–681, 2019.

[7] M. Xu and R. Buyya, "Brownoutcon: A software system based on brownout and containers for energy-efficient cloud computing," *Journal of Systems and Software*, vol. 155, pp. 91–103, 2019.

[8] H. Khazaei, C. Barna, and M. Litoiu, "Performance modeling of microservice platforms," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2848–2862, 2022.

[9] Z. Cai, D. Liu, Y. Lu, and R. Buyya, "Unequal-interval based loosely coupled control method for auto-scaling heterogeneous cloud resources for web applications," *Concurrency and Computation Practice and Experience*, vol. 32, no. 23, pp. 1–16, 2020.

[10] O. Adam, Y. C. Lee, and A. Y. Zomaya, "Stochastic resource provisioning for containerized multi-tier web services in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 2060–2073, 2017.

[11] G. Yu, P. Chen, and Z. Zheng, "Microscaler: Automatic scaling for microservices with an online learning approach," in *IEEE 26th International Conference on Web Services (ICWS)*, 2019, pp. 68–75.

[12] Z. Cai and R. Buyya, "Inverse queuing model based feedback control for elastic container provisioning of web systems in kubernetes," *IEEE Transactions on Computers*, vol. 71, no. 2, pp. 337–348, 2022.

[13] G. Yu, P. Chen, and Z. Zheng, "Microscaler: Cost-effective scaling for microservice applications in the cloud with an online learning approach," *IEEE Transactions on Cloud Computing*, vol. 10, no. 2, pp. 1100–1116, 2022.

[14] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile dynamic provisioning of multi-tier internet applications," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 3, no. 1, pp. 1–39, 2008.

[15] R. Han, M. M. Ghanem, L. Guo, Y. Guo, and M. Osmond, "Enabling cost-aware and adaptive elasticity of multi-tier cloud applications," *Future Generation Computer Systems*, vol. 32, pp. 82–98, 2014.

[16] Y. Lei, Z. Cai, H. Wu, and R. Buyya, "Cloud resource provisioning and bottleneck eliminating for meshed web systems," in *IEEE 13th International Conference on Cloud Computing (CLOUD)*, 2020, pp. 512–516.

[17] H. Wu, Z. Cai, Y. Lei, J. Xu, and R. Buyya, "Adaptive processing rate based container provisioning for meshed micro-services in kubernetes clouds," *CCF Transactions on High Performance Computing*, vol. 4, no. 2, pp. 165–181, 2022.

[18] Y. Shi, J. Huang, X. Zhao, L. Liu, S. Liu, and L. Cui, "Integrating theoretical modeling and experimental measurement for soft resource allocation in multi-tier web systems," in *IEEE 23th International Conference on Web Services (ICWS)*, 2016, pp. 522–529.

[19] J. Bi, H. Yuan, M. Tie, and W. Tan, "Sla-based optimisation of virtualised resource for multi-tier web applications in cloud data centres," *Enterprise Information Systems*, vol. 9, no. 7, pp. 743–767, 2015.

[20] Y. Lei, Z. Cai, X. Li, and R. Buyya, "State space model and queuing network based cloud resource provisioning for meshed web systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 3787–3799, 2022.

[21] X. Zhao, J. Huang, L. Liu, Y. Shi, S. Liu, C. Pu, and L. Cui, "Real-time soft resource allocation in multi-tier web service systems," in *IEEE 24th International Conference on Web Services (ICWS)*, 2017, pp. 492–299.

[22] M. Beltrán, "Automatic provisioning of multi-tier applications in cloud computing environments," *The Journal of Supercomputing*, vol. 71, pp. 2221–2250, 2015.

[23] H. Chen, Q. Wang, B. Palanisamy, and P. Xiong, "Dcm: Dynamic concurrency management for scaling n-tier applications in cloud," in *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2097–2104.

[24] P. Xiong, Z. Wang, S. Malkowski, Q. Wang, D. Jayasinghe, and C. Pu, "Economical and robust provisioning ofn-tier cloud workloads: A multi-level control approach," in *IEEE 31st International Conference on Distributed Computing Systems (ICDCS)*, 2011, pp. 571–580.

[25] D. Jiang, G. Pierre, and C. H. Chi, "Autonomous resource provisioning for multi-service web applications," in *Proceedings of the 19th International Conference on World Wide Web (WWW)*, 2010, pp. 471–480.

[26] K. Salah, K. Elbadawi, and R. Boutaba, "An analytical model for estimating cloud resources of elastic services," *Journal of Network and Systems Management*, vol. 24, pp. 285–308, 2016.

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

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

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

[30] J. Rahman and P. Lama, "Predicting the end-to-end tail latency of containerized microservices in the cloud," in *IEEE 7th International Conference on Cloud Engineering (IC2E)*, 2019, pp. 200–210.

[31] X. Yin, J. Huang, L. Liu, W. He, and L. Cui, "An reinforcement learning approach for allocating software resources," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 14, pp. 1–16, 2021.

[32] W. Iqbal, M. N. Dailey, and D. Carrera, "Unsupervised learning of dynamic resource provisioning policies for cloud-hosted multitier web applications," *IEEE Systems Journal*, vol. 10, no. 4, pp. 1435–1446, 2016.

[33] W. Iqbal, A. Erradi, M. Abdullah, and A. Mahmood, "Predictive auto-scaling of multi-tier applications using performance varying cloud resources," *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 595–607, 2022.

[34] Y. Wang and X. Wang, "Performance-controlled server consolidation for virtualized data centers with multi-tier applications," *Sustainable Computing: Informatics and Systems*, vol. 4, no. 1, pp. 52–65, 2014.

[35] E. Makridis, K. Deliparaschos, E. Kalyvianaki, A. Zolotas, and T. Charalambous, "Robust dynamic cpu resource provisioning in virtualized servers," *IEEE Transactions on Services Computing*, vol. 15, no. 2, pp. 956–969, 2022.

[36] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, "Automated control of multiple virtualized resources," in *Proceedings of the 4th ACM European conference on Computer systems (EuroSys)*, 2009, pp. 13–26.

[37] H. Khazaei, N. Mahmoudi, N. Mahmoudi, and N. Mahmoudi, "Performance modeling of microservice platforms," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2848–2862, 2022.

[38] A. U. Gias, G. Casale, and M. Woodside, "Atom: Model-driven autoscaling for microservices," in *39th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 1994–2004.

[39] E. Suárez, C. M. Pérez, R. Rivera, and M. N. Martínez, *Applications of Regression Models in Epidemiology*. Hoboken, New Jersey, USA: Wiley, 2017.

[40] J. L. Hellerstein, Y. Diao, and S. Parekh, *Feedback control of computing systems*. Hoboken, New Jersey, USA: Wiley, 2004.

[41] Z. Cai, H. Wu, X. Jiang, X. Li, and R. Buyya, "Deep learning and feedback control based container auto-scaling for cloud native micro-services," *IEEE Transactions on Services Computing*, vol. 18, no. 5, pp. 2714–2725, 2025.

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

[43] N. K. S. Center, "Nasa http web server log data," https://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html, 1995, accessed: 2026.