Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **6**, Number **4**, November **2021**

Performance Evaluation of Job Queue Management for a Computer Center with Two Heterogeneous Clusters

Zhixing Lin

[1]Network Center, Sanming University
[2]Institute of Information Engineering, Sanming University
Sanming City, Fujian Province 365004, China
6051751@qq.com

Chibiao Liu

[1]Institute of Information Engineering, Sanming University
[2]Fujian Key Lab of Agriculture IOT Application
[3]IOT Application Engineering Research Center of Fujian Province Colleges and Universities
Sanming City, Fujian Province 365004, China
lcbsmc@163.com

Chia-Hung Wang*

College of Computer Science and Mathematics, Fujian University of Technology
Fuzhou City, Fujian Province 350118, China
*Corresponding Author: jhwang728@hotmail.com

Receive: June 2021; Revise: August 2021

Abstract. *In this paper, we study a job scheduling problem for a computer center with two heterogeneous computing clusters, where one type of clusters offers computing service with a finite-capacity buffer and the other type of clusters offers unlimited waiting buffer. We formulate the job queueing process as a queueing model based on a state-dependent Quasi-Birth-and-Death (QBD) process. Besides, we derive a matrix-analytic method to determine the steady-state probability, and present computing formulas for evaluating the average queue length and waiting time. Through the presented matrix geometric solution, we can obtain the performance measures of interest for the heterogeneous computing system. In the numerical experiments, we also illustrate the relationship between the system performance and model parameters of our interest. It shows that the proposed two-level job queueing system makes full use of the overall resources of the computing platform, and achieves a better job scheduling performance. The findings in this work could reveal the managerial insights into the operational process of computer centers with two-type heterogeneous clusters.*
Keywords: Computing Cluster Management, Job Scheduling, Queueing System, Heterogeneous Service, Matrix-Geometric Method.

1. **Introduction.** In recent years, the operation of a collection of heterogeneous computing servers has become an essential and important managerial issue for the university-level computer centers. As an important base of national science and technology innovation, several colleges and universities in China have established their own service platforms of high-performance computing in order to improve their scientific research strength and level. In the national key special plan, "The 13th Five Year Plan for National Science and

Technology Innovation", it was clearly proposed to develop advanced computing technology, and focus on strengthening the research, development and application of high-performance computing [1]. Various computer simulation applications have been applied to important research fields [2], such as environment and disaster protection, instrument manufacturing, digital contents, biomedical, engineering design, etc.

In the routine work of the university-level computer center, there are many scientific applications that require the execution of a large number of independent jobs resulting in significant overall execution time. Users continually submit their computing jobs to the system, each with unique service-level requirements as well as value to the user and/or resource owner. The implementations of heterogeneous systems in computer centers have been lasting progress for past decades [3-5]. The collection of heterogeneous servers is fed by a single common stream of computing jobs, where each job is dispatched to exactly one type of the heterogeneous servers for processing or computing. The charge of job scheduling is to determine when and how each job should be executed in order to maximize the aggregate utility of our system. Therefore, it is an important issue for managers in the computer center to run their heterogeneous servers efficiently and to be easily accessible for end-users.

For the better understanding of the dynamic behavior of the involved processes, we have to deal with constructions of mathematical models which describe the stochastic service of randomly arriving requests [6-9]. The queueing process of computing jobs scheduled on heterogeneous servers is complicated [10, 11]. Various managerial goals for job assignment, such as queuing efficiency and system utilization, are usually conflicting [12, 13]. The matrix analytic method is one of the most commonly used mathematical tools in queueing theory to evaluate the performance of queueing systems of our interest, such as those works in [14-16] and references therein.

The aim of our work is to provide an analytic mechanism for managing heterogeneous servers in a computer center. We are going to derive a two-tier queueing model with two types of heterogeneous servers, where one type offers the upscale service with a limited buffer size, and the other offers the normal service with unlimited waiting space. In this paper, we will develop a computing method to evaluate the performance of the studied system. We will formulate a two-dimensional state dependent quasi-birth-and-death (QBD) process, and determine the steady-state probability for the studied queueing system. Besides, we will also conduct a queueing analysis to illustrate the impact of the finite buffer on the system performance, such as the average queue length and the average waiting time. The main contribution of this study is to provide a matrix-analytic solution to analyze the stochastic process for the proposed two-tier queueing system.

The structure of the present paper is organized as follows. In Section 2, we will introduce the development of high performance computing platforms in Chinese universities. In Section 3, we are going to derive a two-tier queueing model and demonstrate a queueing analysis based on a matrix-analytic method. In Section 4, we will conduct a series of numerical experiments on the system performance when varying the model parameters. Finally, the concluding remarks are summarized in Section 5.

2. **High Performance Computing Platforms in University-level Computer Centers.**

In this section, we introduce the construction of university-level computer centers, and summarize the applications in scientific research of high-performance computing platform in Chinese universities. In recent years, Chinese universities have paid more and more attention to high-performance computing, and spent a lot of money to establish high-performance computing public service platform. Due to the great progress made by high performance computing in many fields, colleges and universities deeply realize that

high performance computing platform plays an important supporting role in improving the ability of technological innovation, accelerating the speed of innovation, and reducing the research and development cost.

In China, the main mission of high performance computing platforms at colleges and universities is to provide broad computing services to research community, including large-scale scientific MPI applications in various research fields. For example, many scientific applications can be computed on the high performance computing platforms [17, 18], including the design of disaster prevention technology, first principle simulation, biomedical research of molecular dynamics, next generation sequencing analysis, the numerical simulation of weather forecasting, and so on. For dealing with large-scale computer simulations in those works, several simulation software packages could be installed in the clusters at our computer centers, such as WRF, NAMD, LAMMPS, CPMD, SOAPdenovo, Bowtie, ABINIT, etc.

With the rapid development of hardware acceleration technology, several computer centers tend to configure computing devices with different performance in each clusters. For example, in addition to CPU, they would be also equipped with graphic processing units (GPU) and multi-core integrated cores, which results in heterogeneous computing clusters [19, 20]. While greatly improving the performance of the computing platform, the heterogeneous clusters also bring several challenges to the operational management of scheduling computing jobs, such as those works in [21-25].

As a university-level computer center, the computing platform aims to provide scientific computing and simulation services for related disciplines of the whole school. Taking a computer center at Sanming University as an example, the requirements of the computing platform for job scheduling are summarized as follows:

i. It provides a single job management function for users, that is, users submit, query and manage their own serial and parallel jobs through a single job tool.
ii. It makes full use of idle computing resources, improves the overall resource utilization, reduces the job queuing time.
iii. It can ensure the fairness of resource use between user groups.
iv. It can ensure the efficient resource allocation of large parallel jobs, and to achieve efficient scheduling of large-scale computing jobs.
v. A variety of job queues are provided to meet the various computing needs of different users.

The parallel computer systems are valuable resources that are shared commonly by the users in our university. The computing platform at our computer center has become a basic means of teaching and scientific research innovation, which is conducive to the improvement of the comprehensive competitiveness.

The computing power provided by a computer center would greatly improve the ability and level of scientific works, and contribute to the progress of related teaching in colleges and universities. For example, the high performance computing platform at Chinese universities can provide support conditions for the long-term development of the discipline construction of the colleges and universities. Besides, it also improves the scientific research strength and level of Chinese universities. However, there are quite a number of university-level computing platforms with low efficiency, resulting in a waste of resources.

Therefore, it is an important issue for colleges and universities to develop a good management scheme to make these expensive equipment play a full role. In the present paper, we are going to conduct a queueing analysis on job scheduling at a computer center with two-type heterogeneous clusters to improve the effectiveness and use efficiency of the computing platform.

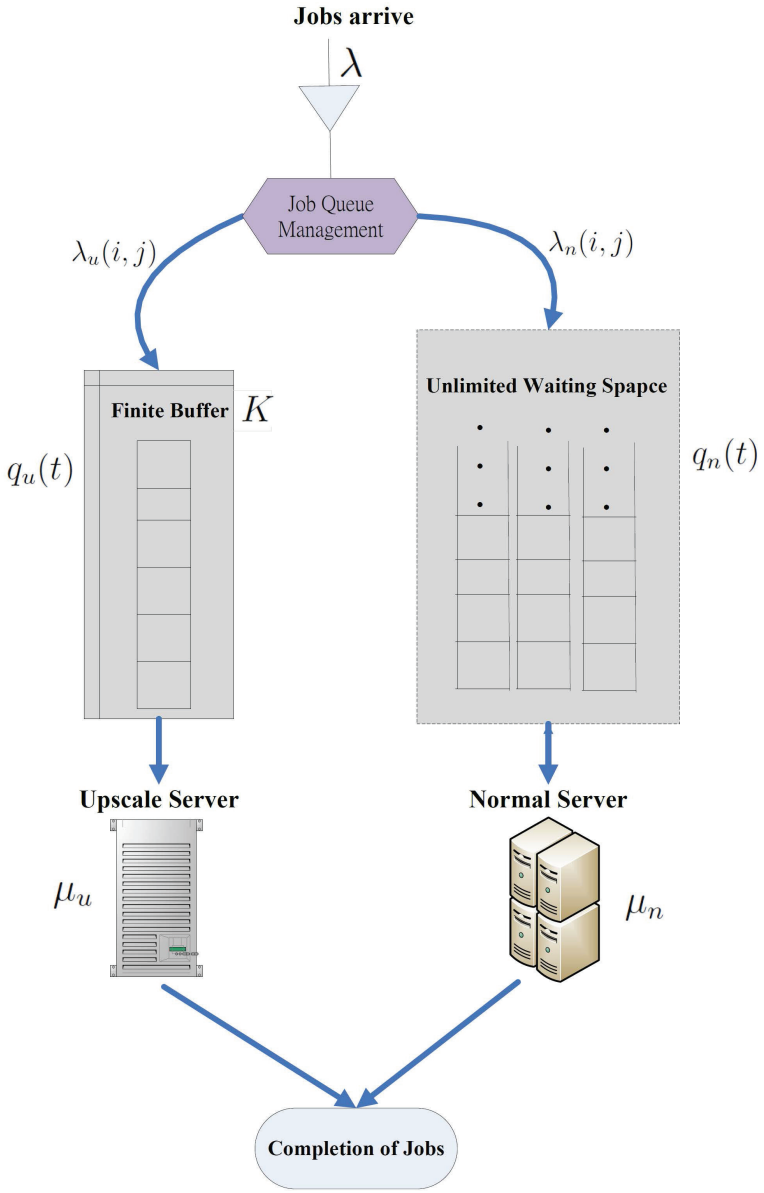

FIGURE 1. A two-tier queueing system with heterogeneous servers.

3. **A Queueing Analysis for Job Management on a Computing Platform with Two Heterogeneous Clusters.** In this section, we introduce a queueing analysis for job management on a computing platform with two heterogeneous clusters. We will formulate a queueing model to analyze the queueing process of jobs to be scheduled on two types of heterogeneous computing clusters. In addition, we will also derive a computing method for determining the system performance.

3.1. **Problem Definitions.** As shown in Figure 1, we study a two-tier queueing system with heterogeneous servers. The job scheduling system consists of a single common login management of jobs and two heterogeneous servers, where the normal server offers service without limited waiting space, and the upscale server offers high-performance service with a finite buffer size K. Note that each arriving job is dispatched to exactly one type of these two heterogeneous servers for executions. The computing jobs dispatched to a particular type of servers are run in First-Come-First-Served order [26].

We assume that the arrival process of jobs occurring at the computing platform follows a Poisson distribution with average rate λ. Meanwhile, it is assumed that the running time of jobs follows exponential distributions. We denote the average service rate for the normal server as μ_n, and denote the average service rate for the upscale server as μ_u.

Next, we denote $q_n(t)$ as the queue length (including the job in service) at the normal server at time t, and denote $q_u(t)$ as the queue length (including the job in service) at the upscale server at time t. Then, the system state of the studied two-tier queueing system can be defined as $(q_n(t), q_u(t))$ on the state space

$$\mathbf{\Omega} = \{(q_n, q_u) |\ q_n \in \{0\} \cup \mathbb{N};\ q_u = 0, 1, \ldots, K\}, \tag{1}$$

where $\mathbb{N}$ is the set of all positive integers. For each state $(i, j) \in \mathbf{\Omega}$, there is a probability that a job is assigned to either the normal server or the upscale server, which results in a state-dependent arrival rate. We denote the state-dependent arrival rate to the normal server as $\lambda_n(i, j)$, and denote the state-dependent arrival rate to the upscale server as $\lambda_u(i, j)$, for each state $(i, j) \in \mathbf{\Omega}$. Therefore, we can formulate the studied two-tier queueing model as a state-dependent QBD process [27]. In Table 1, we summarize all model parameters for the proposed two-tier queueing system with a finite buffer for the upscale server.

Definition 3.1. *The **traffic load** is defined as the fraction of the time in which server is occupied. That is, the average occupancy of the normal server is determined as*

$$\rho_n \triangleq \frac{\overline{\lambda}_n}{\mu_n}, \tag{2}$$

where $\overline{\lambda}_n$ is the effective arrival rate to the normal server. In addition, the average occupancy of the upscale server is determined as

$$\rho_u \triangleq \frac{\overline{\lambda}_u}{\mu_u}, \tag{3}$$

where $\overline{\lambda}_u$ is the effective arrival rate to the upscale server.

Table 1. Notations for Model Parameters

Notation	Description of model parameters
λ	The average arrival rate of jobs to the computing platform.
μ_n	The average service rate for the normal server.
μ_u	The average service rate for the upscale server.
K	The limited capacity of finite buffer for the upscale server.
$q_n(t)$	The number of jobs in service or waiting for the normal server at time t.
$q_u(t)$	The number of jobs in service or waiting for the upscale server at time t.
$\lambda_n(i, j)$	The state-dependent arrival rate to the normal server at state $(i, j) \in \mathbf{\Omega}$.
$\lambda_u(i, j)$	The state-dependent arrival rate to the upscale server at state $(i, j) \in \mathbf{\Omega}$.

3.2. **A Computing Approach to Determine System Performance.** The states of the two-tier queueing system can be classified into three categories: the states represent "all arriving jobs join the normal server", the states represent "jobs would join either the normal server or the upscale server", and the states in region III are "all arriving jobs

join the upscale server". Based on the classification of the states, we can determine the transition rate matrix $\mathbf{Q}$ as follows:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{B}_{0,0} & \mathbf{C}_{0,1} & & & & & & \\ \mathbf{A} & \mathbf{B}_{1,1} & \mathbf{C}_{1,2} & & & & & \\ & \mathbf{A} & \mathbf{B}_{2,2} & \mathbf{C}_{2,3} & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \mathbf{A} & \mathbf{B}_{K,K} & \mathbf{C}_{K,K+1} & & \\ & & & & \mathbf{A} & \mathbf{B} & \mathbf{C} & \\ & & & & & \ddots & \ddots & \ddots \end{bmatrix},$$

where all elements are $(K+1) \times (K+1)$ matrices. The elements of the infinitesimal generator matrix $\mathbf{Q}$ are given according to the transition rate from the state (n, u) to the next state (n', u').

In the infinitesimal generator matrix $\mathbf{Q}$, we have the elements $\mathbf{C}_{0,1} = \lambda \cdot \mathbf{I}$ and $\mathbf{A} = \mu_n \cdot \mathbf{I}$, where $\mathbf{I}$ is the $(K+1) \times (K+1)$ identity matrix. Besides, we have the submatrices

$$\mathbf{C} = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ \vdots & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \lambda \end{bmatrix},$$

and

$$\mathbf{C}_{i,i+1} = \begin{bmatrix} \lambda_n(i,0) & & & & & & \\ & \ddots & & & & & \\ & & \lambda_n(i,i-1) & & & & \\ & & & \lambda & & & \\ & & & & \ddots & & \\ & & & & & \lambda & \end{bmatrix},$$

for $i = 1, \ldots, K$. Meanwhile, we can also determine the submatrices

$$\mathbf{B}_{0,0} = \begin{bmatrix} -\lambda & & & \\ \mu_u & -(\lambda+\mu_u) & & \\ & \ddots & \ddots & \\ & & \mu_u & -(\lambda+\mu_u) \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} -(\mu_n+\lambda) & \lambda & 0 & \cdots & \cdots & \cdots & 0 \\ \mu_u & -(\lambda+\mu_u+\mu_n) & \lambda & 0 & \cdots & \cdots & \vdots \\ 0 & \mu_u & -(\lambda+\mu_u+\mu_n) & \lambda & 0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & 0 & \mu_u & -(\lambda+\mu_u+\mu_n) & \lambda & 0 \\ \vdots & \cdots & \cdots & 0 & \mu_u & -(\lambda+\mu_u+\mu_n) & \lambda \\ 0 & \cdots & \cdots & \cdots & 0 & \mu_u & -(\lambda+\mu_u+\mu_n) \end{bmatrix},$$

and submatrix $\mathbf{B}_{i,i} =$

$$\begin{bmatrix} -(\mu_n+\lambda) & \lambda_u(i,0) & 0 & \cdots & \cdots & \cdots & 0 \\ \mu_u & -(\lambda+\mu_u+\mu_n) & \lambda_u(i,1) & 0 & \cdots & \cdots & \vdots \\ 0 & \mu_u & -(\lambda+\mu_u+\mu_n) & \lambda_u(i,2) & 0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & 0 & \mu_u & -(\lambda+\mu_u+\mu_n) & \lambda_u(i,i-2) & 0 \\ \vdots & \cdots & \cdots & 0 & \mu_u & -(\lambda+\mu_u+\mu_n) & \lambda_u(i,i-1) \\ 0 & \cdots & \cdots & \cdots & 0 & \mu_u & -(\lambda+\mu_u+\mu_n) \end{bmatrix},$$

for all $i = 1, \ldots, K$.

Under the stability condition, we can determine the stationary probability vector

$$\boldsymbol{\pi}_i = [\pi_{i,0}, \pi_{i,1}, ..., \pi_{i,K}], \tag{4}$$

where indices $i = 0, 1, \ldots$, denote the steady states for the normal server. When $i \geq K$, the matrix geometric solution for such a QBD process can be obtained by the following recurrence relation

$$\boldsymbol{\pi}_{i+1} = \boldsymbol{\pi}_i \mathbf{R}, \tag{5}$$

where matrix $\mathbf{R}$ is the so-called rate matrix used in Matrix-Geometric Method [28, 29].

For $0 \leq i \leq K$, the probability vector $\boldsymbol{\pi}_i$ can be obtained by solving a set of equations. From $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$ and (5), the steady-state vectors $\boldsymbol{\pi}_0$, $\boldsymbol{\pi}_1$, ..., and $\boldsymbol{\pi}_K$ can be solved from the boundary conditions and the normalization condition:

$$\begin{aligned} \boldsymbol{\pi}_0\mathbf{B}_{0,0} + \boldsymbol{\pi}_1\mathbf{A} &= \mathbf{0}, & (6)\\ \boldsymbol{\pi}_0\mathbf{C}_{0,1} + \boldsymbol{\pi}_1\mathbf{B}_{1,1} + \boldsymbol{\pi}_2\mathbf{A} &= \mathbf{0}, & (7)\\ \boldsymbol{\pi}_1\mathbf{C}_{1,2} + \boldsymbol{\pi}_2\mathbf{B}_{2,2} + \boldsymbol{\pi}_3\mathbf{A} &= \mathbf{0}, & (8)\\ &\vdots & \\ \boldsymbol{\pi}_{K-2}\mathbf{C}_{K-2,K-1} + \boldsymbol{\pi}_{K-1}\mathbf{B}_{K-1,K-1} + \boldsymbol{\pi}_K\mathbf{A} &= \mathbf{0}, & (9)\\ \boldsymbol{\pi}_{K-1}\mathbf{C}_{K-1,K} + \boldsymbol{\pi}_K(\mathbf{B}_{K,K} + \mathbf{R}\mathbf{A}) &= \mathbf{0}, & (10)\\ \boldsymbol{\pi}_0\mathbf{1} + \boldsymbol{\pi}_1\mathbf{1} + \cdots + \boldsymbol{\pi}_K(\mathbf{I}-\mathbf{R})^{-1}\mathbf{1} &= 1, & (11) \end{aligned}$$

where $\mathbf{1}$ is the column vector of ones. After the stationary distribution is determined, we can obtain the major system performance measures, i.e., the average queue length, the average waiting time, and the effective arrival rate to each type of server.

Proposition 3.1. *The average queue length of jobs waiting for the normal server can be determined as*

$$L_n = \sum_{i=0}^{\infty} i \cdot (\boldsymbol{\pi}_i \cdot \mathbf{1}) = \sum_{i=0}^{\infty}\sum_{j=0}^{K} i \cdot \pi_{i,j}, \tag{12}$$

and the average queue length of jobs waiting for the upscale server is determined as

$$L_u = \sum_{j=0}^{K} j \cdot (\sum_{i=0}^{\infty} \pi_{i,j}) = \sum_{j=0}^{K}\sum_{i=0}^{\infty} j \cdot \pi_{i,j}, \tag{13}$$

where $\mathbf{1}$ *is the column vector of ones.*

Besides, we can estimate the average ratio of arriving jobs assigned to each type of server by means of the effective arrival rates as follows.

Proposition 3.2. *The effective arrival rate to the normal server can be determined as*

$$\overline{\lambda}_n = \sum_{i=0}^{\infty} \sum_{j=0}^{K} \lambda_n(i,j) \cdot \pi_{i,j}, \tag{14}$$

and the effective arrival rate to the upscale server is determined as

$$\overline{\lambda}_u = \sum_{i=0}^{\infty} \sum_{j=0}^{K} \lambda_u(i,j) \cdot \pi_{i,j}. \tag{15}$$

According to Little's formula [27], we can estimate the average waiting time for each type of server by means of the following equation (16) and equation (17) .

Proposition 3.3. *We compute the average waiting time for the normal server via*

$$W_n = \frac{L_n}{\overline{\lambda}_n}, \tag{16}$$

where $\overline{\lambda}_n$ is the effective arrival rate to the normal server. Similarly, the average waiting time for the upscale server can be determined as

$$W_u = \frac{L_u}{\overline{\lambda}_u}, \tag{17}$$

where $\overline{\lambda}_u$ is the effective arrival rate to the upscale server.

In summary, we implement the following algorithm to determine the system performance of the proposed two-tier queueing system, such as the average queue length and the average waiting time.

***A Algorithm for Determining System Performance:*:**

Input: The average arrival rate λ, average service rate μ_n for the normal server, and average service rate μ_u for the upscale server .

Step 1.: We determine the transition rate matrix $\mathbf{Q}$ and its submatrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{B}_{i,i}$, and $\mathbf{C}_{i,i+1}$, for all $i = 1, \ldots, K$.

Step 2.: We estimate the rate matrix $\mathbf{R}$ through $\mathbf{R}^2\mathbf{A} + \mathbf{R}\mathbf{B} + \mathbf{C} = \mathbf{0}$.

Step 3.: We solve the system of linear equations $\mathbf{R}^2\mathbf{A}+\mathbf{R}\mathbf{B}+\mathbf{C} = \mathbf{0}$ and normalization condition $\boldsymbol{\pi}_0\mathbf{1} + \boldsymbol{\pi}_1\mathbf{1} + \cdots + \boldsymbol{\pi}_K(\mathbf{I} - \mathbf{R})^{-1}\mathbf{1} = 1$.

Step 4.: We obtain the stationary distribution $\boldsymbol{\pi}_0$, $\boldsymbol{\pi}_1$, ..., and $\boldsymbol{\pi}_K$.

Output: The average queue length L_n, L_u, the effective arrival rate $\overline{\lambda}_n$, $\overline{\lambda}_u$, and the average waiting time W_n, W_u.

4. **Numerical Results.** In this section, we conduct a series of numerical experiments to figure out the applicability of the proposed two-tier queueing model. We use the computing language Matlab to compile the proposed computing method, and the numerical experiments are conducted on the PC platform with Windows 10 Professional 64-bit with the processor Intel (R) Core (TM) i7-6500U CPU@2.50GHz dual-core and 32 GB memory.

The parameter settings for the two-tier queueing model are given as follows. As an illustrative example, we take the average service rate $\mu_n = 0.6$ (jobs/hour) for the normal server, and take the average service rate $\mu_u = 0.7$ (jobs/hour) for the upscale server. In our numerical experiments, three load scenarios are also taken into consideration to compare the system performance. Those three traffic load scenarios include the heavy load scenario when average arrival rate is $\lambda = 1.0$ (jobs/hour), the median load scenario when average arrival rate is $\lambda = 0.7$ (jobs/hour), and the light load scenario when average arrival rate is $\lambda = 0.5$ (jobs/hour).

In Figures 2-5, we illustrate the numerical results obtained by presented matrix-geometric method when we vary the finite buffer size K from 2 to 15. The red line represents a heavier traffic load, and the green line indicates a lower traffic load. Meanwhile, the blue curve represents the scenario when the traffic load is between them. With the stationary distribution $\boldsymbol{\pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots)$, we can determine the expected number of jobs waiting for the normal server, L_n, and the expected number of jobs waiting for the upscale server, L_u. By means of equations (14) and (15), we also can compute the effective arrival rates $\overline{\lambda}_n$ and $\overline{\lambda}_u$ for each type of server, individually. Besides, we can determine the average waiting time for the normal server, W_n, and the average waiting time for the upscale server, W_u, by using the formulas (16) and (17).

In Figure 2, it indicates the expected numbers of jobs waiting for each server, L_n and L_u, individually. When we increase the buffer size K, it is found that the average queue length L_n of the normal server is decreasing, while the average queue length L_u of the upscale server is increasing. We also find that the marginal benefit of increasing the buffer size would gradually decrease as it increases to a sufficient level.

In Figure 3, it illustrates the average ratio of arrivals to be assigned to each type of server. Those effective arrival rates, $\overline{\lambda}_n$ and $\overline{\lambda}_u$, are depicted under three traffic load scenarios: heavy traffic load, median traffic load, and light traffic load. We find that it relies more heavily on the normal server when the traffic load is lighter. But, when the traffic load is severer, the reliance on the upscale server will increase.

In Figure 4, we show the impact of increasing the buffer capacity on the average waiting time for each type of server, respectively. As the buffer size K increases from 2 to 15, it shows a decreasing situation for the average waiting time W_n for the normal server, while it shows an increasing situation for the average waiting time W_u of the upscale server. When the buffer capacity increases to a sufficient level, the marginal benefit of improving the average waiting time would gradually decrease.

In Figure 5, we find that the running time (CPU time) needed by the presented matrix-geometric method would become more time consuming as the buffer size K grows large. In the large-scale case with huge size K, it would result in a large number of boundary states and a large number of phases of the QBD process, which would greatly increase the computational complexity and may cause the ill-conditioned matrices of the traditional iterative algorithm for the rate matrix.

5. **Conclusions.** In this paper, we carried out an operational management for job scheduling at a computer center to improve the service standard of computing platform. We presented a queueing model based on a state-dependent QBD process, and developed a computing method for evaluating the system performance of scheduling job at the computing platform. Furthermore, we conducted a queueing analysis for assigning computing jobs to heterogeneous computer system with a finite-buffer control. In the numerical experiments, it illustrated that a proper buffer control could balance queuing efficiency and the cost of increasing the buffer size. We developed a management scheme by assigning those arriving jobs to appropriate type of servers under finite buffer control, which achieved better system performance.

In the practical application, it shows that the proposed two-tier job scheduling system makes full use of the overall resources of the computing platform, and performs better job scheduling. By means of optimizing the operational and managerial processes, those non-standard operations that would affect the overall performance of the system could be avoided, which could be helpful in reducing the additional system overhead and resource waste at the computer centers.

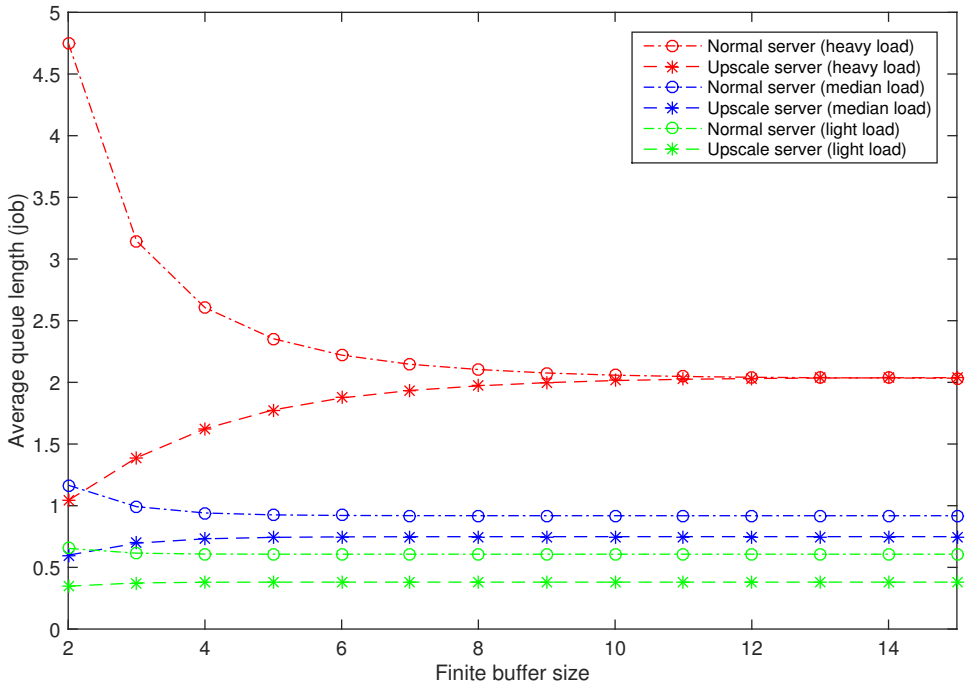

FIGURE 2. The average queue size (L_n and L_u) versus the finite buffer size K under three load scenarios.

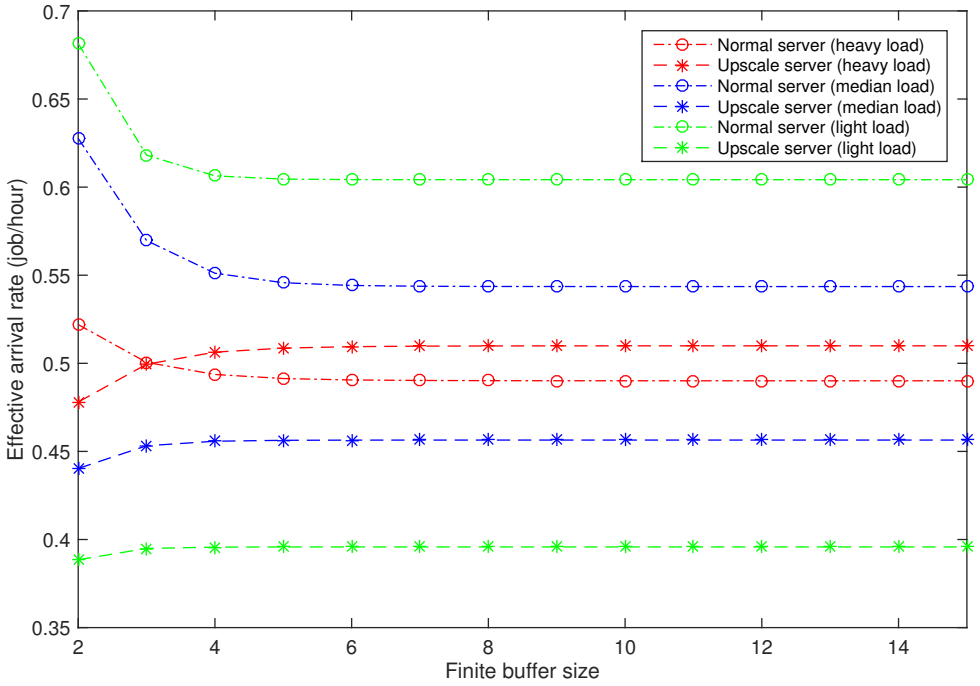

FIGURE 3. The effective arrival rate ($\overline{\lambda}_n$ and $\overline{\lambda}_u$) versus the finite buffer size K under three load scenarios.

Moreover, several suggestions on the management and operation of a computer center at the colleges and universities are summarized as follows. First of all, it is essential to

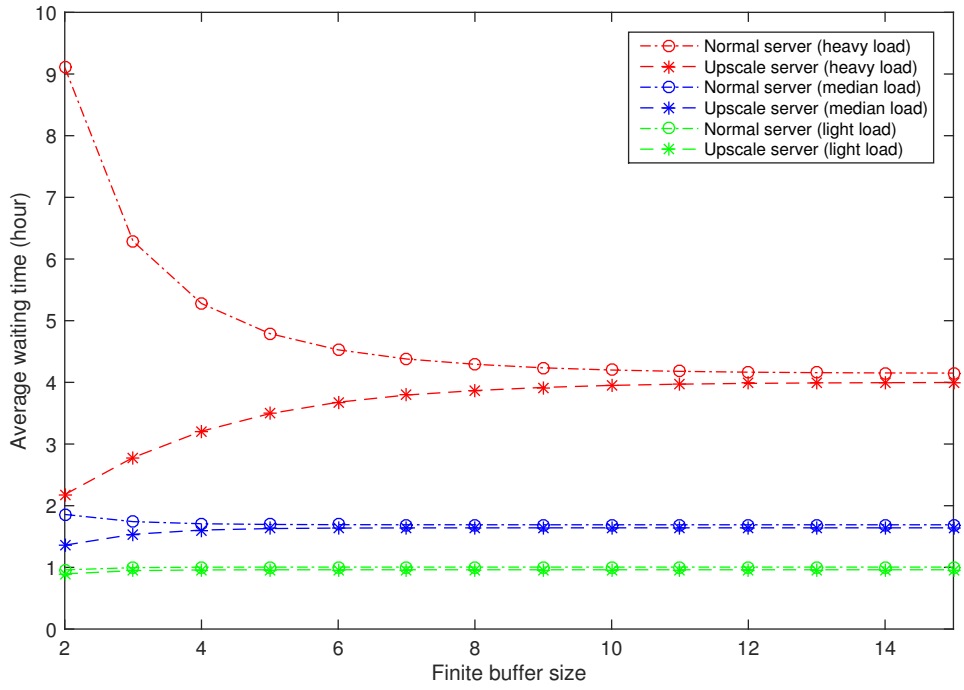

FIGURE 4. The average waiting time (W_n and W_u) versus the finite buffer size K under three load scenarios.

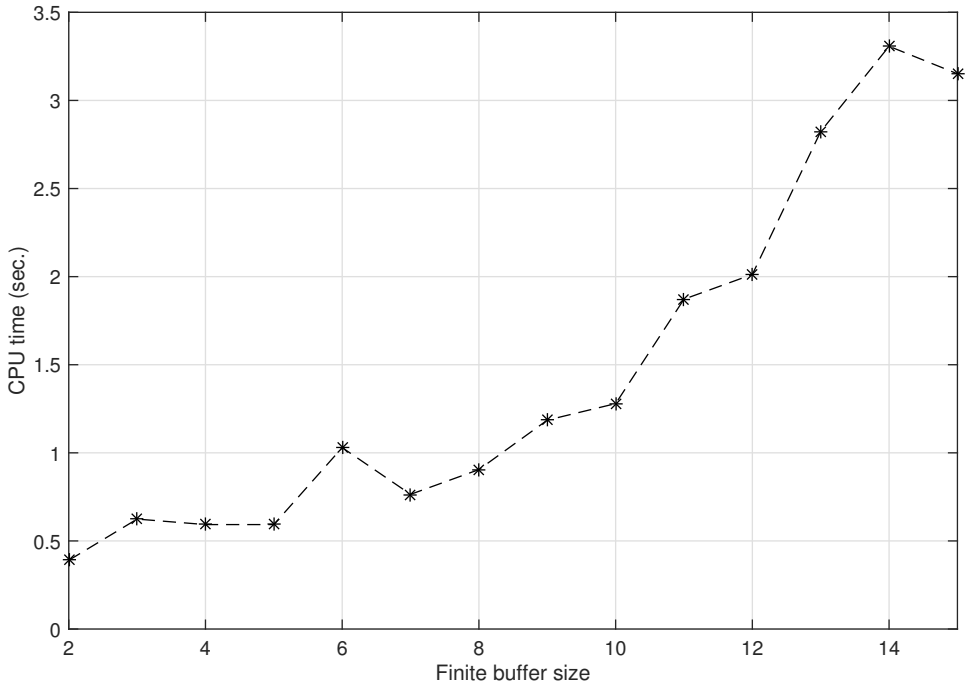

FIGURE 5. The CPU time versus the finite buffer size K.

organize a professional management and operation team. A professional technical team is needed at the computer center for software application development, hardware operation and maintenance, especially in the expansion and development of software and application,

the management and scheduling of resources, the optimization of calculation model and method. Secondly, in order to improve the efficiency of the platform, we should introduce the hot spots in the current research field into the campus, such as Exascale computing, quantum computing, intelligent computing, machine learning, and so on. Finally, we can carry out the training and lectures to help our users know well the computing platform, which would save more time for operators and managers at the computer center.

Acknowledgment. This work is supported by Digital Fujian Research Institute for Industrial Energy Big Data, Fujian Province University Key Laboratory for the Analysis and Application of Industry Big Data, Fujian Key Lab of Agriculture IOT Application, and IOT Application Engineering Research Center of Fujian Province Colleges and Universities under grant numbers No. S202011311059X, No. 2020-S-11, and No. 202002064034.

REFERENCES

[1] The State Council of the People's Republic of China, The 13th Five Year Plan for National Science and Technology Innovation. Accessed on July 28, 2016. Available at `http://www.gov.cn/`

[2] A. Mohammed, A. Eleliemy, F.M. Ciorba, F. Kasielke, and I. Banicescu, An approach for realistically simulating the performance of scientific applications on high performance computing systems, *Future Generation Computer Systems*, vol. 111, pp. 617–633, 2020.

[3] M.A. Schaar and K. Efe, Effective queueing strategies for co-scheduling in a pool of processors, *Computer Communications*, vol. 19, pp. 743–753, 1996.

[4] B. Arafeh, K. Day, and A. Touzene, A multilevel partitioning approach for efficient tasks allocation in heterogeneous distributed systems, *Journal of Systems Architecture*, vol. 54, no. 5, pp. 530–548, 2008.

[5] J.M. Rivas, J.J. Gutiérrez, and M.G. Harbour, A supercomputing framework for the evaluation of real-time analysis and optimization techniques, *Journal of Systems and Software*, vol. 124, pp. 120–136, 2017.

[6] C.-H. Wang, A modelling framework for managing risk-based checkpoint screening systems with two-type inspection queues, in: *Proceedings of the Third International Conference on Robot, Vision and Signal Processing (RVSP 2015)*, IEEE, pp. 220–223, Kaohsiung, Taiwan, November 18-20, 2015.

[7] C.-H. Wang, Arena simulation for aviation passenger security-check systems, in: *Advances in Intelligent Systems and Computing*, J. Pan et al. (eds.), vol. 536, Chapter 12, pp. 95–102, 2016.

[8] C.-H. Wang, A three-level health inspection queue based on risk screening management mechanism for post-COVID global economic recovery, *IEEE Access*, vol. 8, pp. 177604–177614, 2020.

[9] J. Doncel, V. Mancuso, Optimal performance of parallel-server systems with job size prediction errors, *Operations Research Letters*, vol. 49, pp. 459–464, 2021.

[10] A. Davydow, P. Chuprikov, S.I. Nikolenko, and K. Kogan, Competitive buffer management for packets with latency constraints, *Computer Networks*, vol. 189, 107942, 2021.

[11] Z. Scully, I. Grosof, and M. Harchol-Balter, Optimal multiserver scheduling with unknown job sizes in heavy traffic, *Performance Evaluation*, vol. 145, 102150, 2021.

[12] C.-H. Wang and X.J. Wu, Performance analysis of a security-check system with four types of inspection channels for high-speed rail stations in China, in: *Smart Service Systems, Operations Management, and Analytics*, H. Yang et al. (eds.), 2019 INFORMS International Conference on Service Science (ICSS2019), Nanjing, China, June 27-29, 2019. *Springer Proceedings in Business and Economics*, pp. 7–16. 2020.

[13] C.-H. Wang, Y.-T. Chen, and X.J. Wu, A multi-tier inspection queueing system with finite capacity for differentiated border control measures, *IEEE Access*, vol. 9, pp. 60489–60502, 2021.

[14] H. Luh, Z. Zhang, and C.-H. Wang, A computing approach to two competing services with a finite buffer effect, in:*Proceedings of the 8th International Conference on Queueing Theory and Network Applications*, ACM, pp. 15–21, 2013.

[15] C.-H. Wang, A matrix-analytic method for evaluating performance of security-check system with two-type inspection queues, in: *Proceedings of the 2019 International Conference on Industrial Engineering and Systems Management (IESM 2019)*, IEEE, Shanghai, China, September 25-27, 2019.

[16] Q.G. Zhao, C.-H. Wang, Z.Y. Dong, S.M. Chen, Q.P. Yang, and Y. Wei, A matrix-analytic solution to three-level multi-server queueing model with a shared finite buffer, in: *Proceedings of the 14th International Conference on Genetic and Evolutionary Computing (ICGEC 2021)*, Jilin City, Jilin

Province, China, October 21-23, 2021. *Lecture Notes in Electrical Engineering*, Springer Nature, 2021.

[17] B. Schroeder and M. Harchol-Balter, Evaluation of task assignment policies for supercomputing servers: The case for load unbalancing and fairness, *Cluster Computing*, vol. 7, pp. 151–161, 2004.

[18] D. Sánchez, D. Isern, Á. Rodríguez-Rozas, and A. Moreno, Agent-based platform to support the execution of parallel tasks, *Expert Systems with Applications*, vol. 38, no. 6, pp. 6644–6656, 2011.

[19] S. Soner, and C. ́Ozturan, Integer programming based heterogeneous CPU–GPU cluster schedulers for SLURM resource manager, *Journal of Computer and System Sciences*, vol. 81, no. 1, pp. 38–56, 2015.

[20] A.I. Orhean, F. Pop, and I. Raicu, New scheduling approach using reinforcement learning for heterogeneous distributed systems, *Journal of Parallel and Distributed Computing*, vol. 117, pp. 292–302, 2018.

[21] X. Qin and H. Jiang, A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters, *Journal of Parallel and Distributed Computing*, vol. 65, no. 8, pp. 885–900, 2005.

[22] B. Ucar, C. Aykanat, K. Kaya, and M. Ikinci, Task assignment in heterogeneous computing systems, *Journal of Parallel and Distributed Computing*, vol. 66, no. 1, pp. 32–46, 2006.

[23] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, and K. Li, Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster, *Information Sciences*, vol. 319, pp. 113–131, 2015.

[24] C. Witt, M. Bux, W. Gusew, and U. Leser, Predictive performance modeling for distributed batch processing using black box monitoring and machine learning, *Information Systems*, vol. 82, pp. 33–52, 2019.

[25] S.M. MirhoseiniNejad, G. Badawy, and D.G. Down, Holistic thermal-aware workload management and infrastructure control for heterogeneous data centers using machine learning, *Future Generation Computer Systems*, vol. 118, pp. 208–218, 2021.

[26] C.-H. Wang and H. Luh, Estimating the loss probability under heavy traffic conditions, *Computers and Mathematics with Applications*, vol. 64, no. 5, pp. 1352–1363, 2012.

[27] D, Gross and C.M. Harris, *Fundamentals of Queueing Theory*, 3rd ed., John Wiley & Sons, Inc., New York, 1998.

[28] M.F. Neuts, *Matrix-Geometric Solutions in Stochastic Models*. The John Hopkins University Press, 1981.

[29] G. Latouche and V. Ramaswami, *Introduction to Matrix Analytic Methods in Stochastic Modelling*, ASA & SIAM, Philadelphia, USA, 1999.