Hindawi
Security and Communication Networks
Volume 2020, Article ID 8886640, 12 pages
https://doi.org/10.1155/2020/8886640

Research Article

Dynamic Security Exchange Scheduling Model for Business Workflow Based on Queuing Theory in Cloud Computing

Rongbin Xu,[1,2] Jianguo Wu,[2] Yongliang Cheng,[2] Zhiqiang Liu,[1] Yuanmo Lin,[1] and Ying Xie[1]

[1] *College of Information Engineering, Putian University, Putian, China*
[2] *School of Computer Science and Technology, Anhui University, Hefei, China*

Correspondence should be addressed to Ying Xie; xieying@ahu.edu.cn

Received 3 March 2020; Revised 11 June 2020; Accepted 31 July 2020; Published 28 August 2020

Academic Editor: Benjamin Aziz

With the rapid development of e-business, large volume of business processes need to be handled in a constrained time. There is always a security issue related to on-time completion in many applications in the economic fields. So, how to effectively manage and organize business processes became very important. By using cloud computing, instance-intensive processes can be handled more effectively by applying just-right virtual machines. Hence, the management of cloud resources became an important issue that many researchers focus on to fully utilize the advantage of cloud. In this paper, we mainly discuss the queuing theory and put forward our novel dynamic process scheduling model based on queuing theory, which is named *M/G/k/l-P* for business processes. This model can solve the issue of allocating appropriate number of cloud resources based on the number of tasks and execution stages to ensure whether the numbers of cloud resources are sufficient and adequate or not, which can improve the security issue for business process. The service discipline in our model can provide a dynamic process by setting different priorities to improve the experience of users. Evaluations prove that the queuing model of *M/G/k/l-P* can work very well for business workflow scheduling.

1. Introduction

Business processes are the series of interactions between businesses and their users, vendors, and other related partners [1]. A business workflow is an instance of well-defined business process that is often repeated as a part of standard enterprise operations [2]. Business process occurs at all organizational levels and may not be visible to the customers. A business process may often be visualized or modeled as a flowchart of a sequence of activities with interleaving decision points, which can also be viewed as a process matrix of a sequence of activities with relevance rules based on data in the process [3]. The benefits of using business processes mainly concentrate on improving the satisfaction of customer and improving agility for reacting to rapid market change [4, 5]. With intensified globalization, e-business is in the process of rapid growth, which often involves processing large numbers of instance-intensive business processes within a constrained period of time [6].

A typical motivating instance-intensive business example is security exchange in the stock market that involves a large number of transactions between different organizations, and each of them is a relatively short workflow instance with only a few steps [7]. A stock depository and clearing corporation in the stock market need to check and produce money transfer details in the night-time for each client who makes deals during the daytime before corresponding money transfers can be processed between the clearing corporation and the designated banks. The requiring allocation of cloud and edge resources is a typical dynamic process depending on the number of transactions [8]. The clearing process and transferring money in stock market may need to be finished before a deadline, for example, 3 : 00 am each weekday [9].

By utilizing the novel infrastructure of cloud and edge computing, instance-intensive processes can be solved easily to apply enough virtual machines for the elastic property of cloud environment [10, 11]. All the users can access those services for running their software applications in pay-as-you-go fashion to avoid huge capital investment, energy consumption, and system maintenance [12]. However, at the same time of meeting the temporal requirement, for examole, deadline of business workflow [13], cost is another factor we should focus on. The proper allocation of cloud resources can improve the efficiency of virtual machines so that there is no waste [14, 15]. In this paper, we apply queuing model for the allocation of cloud resources based on the process of dynamic management, which can be regarded as a very efficient method for cloud allocation. As demonstrated in our evaluation, the queuing model of *M*/*G*/*k*/*l*-*P* is an effective solution for business workflow scheduling.

There are mainly three contributions in our paper.

First, it is the first time that we employ queuing theory for business workflow scheduling. There are a large number of tasks that need to be scheduled in business workflows, which are convenient to operate virtual machines as servers by queuing theory in cloud environment.

Second, our model *M*/*G*/*k*/*l*-*P* applies cloud resources in a dynamic process, which can dynamically manage the number of virtual machines based on service time distribution so there is no much waste.

Third, the service discipline in our model *M*/*G*/*k*/*l*-*P* can significantly improve the Quality of Service (QoS) in business processes. We apply two service disciplines besides the default one in our model so that the tasks in business processes with high priority can be served first based on the different demands of users.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 proposes a motivating example of time-constrained instance-intensive business processes in stock market and gives the detailed problem analysis. Section 4 presents some preliminary contents of queuing theory. Section 5 discusses some classical queuing models and puts forward our novel queuing model of *M*/*G*/*k*/*l*-*P* for business processes. Section 6 provides a set of results for evaluation of our queuing model. Finally, section 7 concludes our contributions and points out the future work.

2. Related Work

Matching abundant business tasks to machines and scheduling the execution order of these tasks are referred to mapping. This problem of mapping has been proved to be an NP-complete issue in [15–17]. A shortest tree algorithm is described in [15] that minimizes the sum of execution and communication costs for arbitrarily connected distributed systems with arbitrary numbers of processors. This algorithm uses a dynamic programming approach to solve the problem for n tasks and m processors in $O(m^2n)$ time. A scheduling risk assessment framework [16] is developed to model the uncertainties in duration and observe their impact on project objectives such as completion time and cost. It provides some important propositions or guidelines for project management practitioners. An Analytics-as-a-Service (AaaS) platform [17] is proposed to deliver on-demand services at low cost in an easy use manner. In this paper, we propose a model that effectively admits data analytics requests, dynamically provisions resources, and maximizes profit for AaaS providers, while satisfying QoS requirements of queries with Service Level Agreement guarantees. It can enhance profits, reduce resource costs, increase query admission rates, and decrease query response times. All these algorithms mainly focus on minimizing execution and communication costs. However, these algorithms have not fully considered the specific scenarios with large amounts of business workflow tasks. In our paper, queuing theory is employed for scheduling large number of business workflow tasks so it is convenient to operate cloud servers for no waste of resources.

In cloud environments, scheduling decisions can be made in the shortest time which are possible for utilizing a distributed suite of different high-performance machines. Cloud computing can offer powerful, on-demand, and elastic computing resources, which is an ideal hosting environment for running a large batch of parallel business processes [18]. Many algorithms have been proposed to schedule the business workflow applications in heterogeneous distributed system environments. Reference [19] is devoted to improving the performance for cloud service platforms by minimizing uncertainty propagation in scheduling workflow applications that have both uncertain task execution time and data transfer time. Moreover, a novel scheduling architecture is designed to control the count of workflow tasks directly waiting on each service instance. Yu et al. [20] propose several challenges for scheduling workflow applications in grid environment and some are hard to resolve; for example, the grid resources are not under the control of the scheduler. They also classify workflow scheduling into two major types: best-effort-based and QoS-constraint-based scheduling. However, these algorithms do not fully utilize the dynamic characteristic of cloud resources. In our paper, the allocation of cloud resources can be a dynamic process based on the initialized number of tasks and execution stage. First, the execution processes are prioritized by different demands of users. Then, our *M*/*G*/*k*/*l*-*P* can dynamically manage the usage of virtual machines based on service time distribution.

There are some existing researches which focus on dynamic resource allocation. They mostly draw attention on energy consumption by using multiple virtual machines and make great contributions to computer science. Reference [21] proposes a cloud resource allocation model based on an imperfect information Stackelberg game (CSAM-IISG) using a hidden Markov model in cloud computing environment. This strategy increases the profit of both the resource suppliers and applicants. Reference [22] presents a queuing model that buffers the same type of VM jobs in one virtual queue. The queuing model then divides the VM scheduling into two parallel low-complexity algorithms, that is, intraqueue buffering and interqueue scheduling. This model can achieve low delay performance in terms of average job completion time and high throughput performance in terms of job hosting ratio. Reference [23] provides a QoS-metric-based resource provisioning technique that can cater to provisioned resource distribution and scheduling of resources. This technique is efficient in reducing execution time and cost of cloud workloads along with other QoS parameters. Reference [24] uses virtualization technology to allocate data center resources dynamically based on application demands and supports green computing by optimizing the number of servers. Waiting time is considered as an important parameter for the evaluation of queuing theory. This strategy combines different types of workloads nicely and improves the overall utilization of server resources. All these strategies have set up a more specific situation and solve the problem of resource allocation. However, these scheduling strategies almost have not considered service discipline to prove high QoS. The queuing model in our *M*/*G*/*k*/*l*-*P* can reduce the number of the virtual machines and significantly improve the QoS in two aspects: dynamically manage the usage of virtual machines and set different service disciplines for the demands of users.

3. Motivating Business Workflow Example and Existing Problem

Security exchange is a typical time-constrained commercial event and any failures of the on-time completion for money transfer in stock market may cause huge financial losses because unsuccessful timely money transfer could result in the failure of making deals, which is definitely a disastrous situation in the stock market. Security exchange is also a typical multistep process and most steps are executed in parallel. For the clearing process, there are six major stages which contain some steps as shown in Figure 1.

For an example of security organization in China, there are tens of thousands of users and their information include account, password, transaction, balance of account, and so on. All these involve threat protection, encryption, data protection, and archiving.

More than one hundred security corporations that may have a great number of branches in security exchange market, which is a typical instance-intensive business process. Customers may choose a certain branch to deal with the security transactions, so the system of security exchange is extremely complex. Figure 1 is just a very simple model that we use to interpret our business workflow scheduling processes. Step 1 in Figure 1 may involve various validation processes and they all log on security stock trading center to conduct transactions. Millions of client entrustments can be requested by clients all over the country. All of them are processed concurrently in more than 4000 branches [25]. Steps 2 and 3 check the raw entrustment data that clients make deals and generate the balance of trades [26]. However, after settling all the trading processes during trading day before 3 : 00 pm (closing time), the real fund settlements should be cleared within three levels as shown in steps 4–6, which are the most important steps for security trading. The first level clearing is between clearing corporations and security corporations, the second one is between security corporations and their branches, and then the third one is between branches and users. These three steps involve a large number of capital flow and money transfer in stock market that need to be finished before 3 : 00 am of each weekday, which are typical time-constrained tasks. Any failures of on-time completion for money transfer in stock market may cause huge financial losses because unsuccessful timely money transfer could result in the failure of making deals, which is definitely a disastrous situation in the security exchange market. The final step, that is, step 6, is to produce clearing files. Security corporations and designated banks should produce the clearing files for clearing corporation. The balance of all transferred capital should be set to zero at the clearing corporation level [27].

As shown in Figure 1, the six steps have a sequential relationship. There is a large number of works to schedule all the security exchange transactions in limited constrained time. The amount of transactions may be much bigger during some special days than common time, like the first day and the weekend of a week, and there are more tasks which should be scheduled in the night. Meanwhile, the multistep transactions are always short-duration activities. The execution time of short-duration activities is normally much smaller than traditional scientific long-duration activities and every activity can be handled and responded in a short time. Based on all these characteristics in stock market, it is well suited to apply cloud environment to schedule security exchange transactions because cloud computing can offer on-demand, elastic, and cost-effective resource as shown in Figure 2.

However, due to the dynamic characteristics of cloud computing, how to allocate cloud resources dynamically in the initiation and execution stages so that the cloud resources can be efficiently utilized without much waste is an extremely complex issue because the allocation of cloud resources should be dynamically adjusted based on the arrival tasks. First, what we need to do is allocating appropriate number of cloud resources based on the initialized number of tasks. Then, execution process of business workflow should be monitored in real time to ensure that the number of cloud resources is sufficient and adequate. It is really a dynamic usage process of cloud resources. In this paper, we apply queuing model *M*/*G*/*k*/*l*-*P* for the allocation of cloud resources. The novel queuing model can be proved as a very efficient method based on our evaluations. Furthermore, we apply different service disciplines in our queuing model, so that it can offer a dynamic scheduling process by setting different priorities to the tasks based on the requirements of users.

4. Preliminary of Queuing Theory

Queuing theory is a mathematical research of waiting lines or queues which focuses on identifying and managing the response time of users for services. Queuing theory was created to describe the Copenhagen telephone exchange originally and the ideas had seen applications including telecommunication [28], traffic engineering [29], computing and the design of factories [30], shops, offices, and hospitals [31]. The theory allows cloud system to be scaled optimally to guarantee the QoS for response time. It can also plan proper deployment and removal of virtual machines according to the system load [32]. So, it is very applicable to be used for our instance-intensive workflow scheduling, which also aims

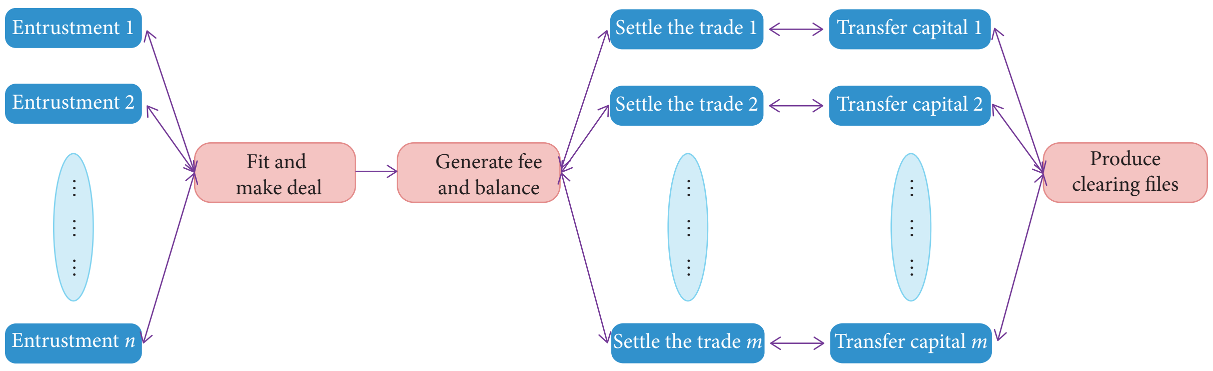

FIGURE 1: Simplified flowchart of multistep process of security exchange.

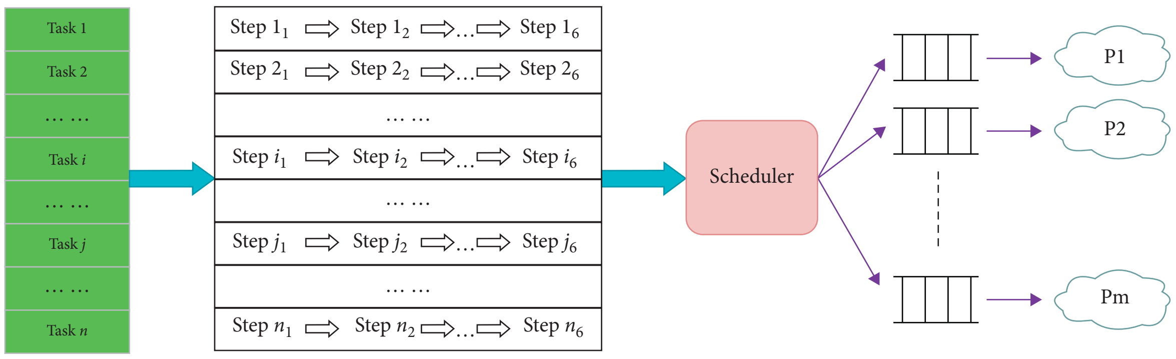
Task 1
Task 2
Task i
Task j
Task n
Step 1_1 ⇒ Step 1_2 ⇒…⇒ Step 1_6
Step 2_1 ⇒ Step 2_2 ⇒…⇒ Step 2_6
Step i_1 ⇒ Step i_2 ⇒…⇒ Step i_6
Step j_1 ⇒ Step j_2 ⇒…⇒ Step j_6
Step n_1 ⇒ Step n_2 ⇒…⇒ Step n_6
Scheduler
P1
P2
Pm

FIGURE 2: Scheduling model for a batch of parallel business tasks.

to get an excellent scheduling result in the response time of various kinds of tasks dynamically. The queuing model is constructed in advance so that the queue length and waiting time can be predicted by queuing theory (https://en.wikipedia.org/wiki/Queueing_theory-cite_note-sun-1).It is generally considered as a branch of process management research because the results are often used when making business decisions about the required resources to provide a good service.

Queuing systems can be characterized by various probabilistic properties, such as complex input process, service time distribution, number of servers, buffer size, and queue discipline. These properties can be described as shown in Figure 3, which is the fundamental form for different kinds of queuing theories.

The aim of investigations in queuing system is to get the main measures of the system, which are the probabilistic properties of following random variables: number of tasks in the system, number of waiting tasks in each server, response time of tasks, waiting time of a task, and utilization of all servers. To fully utilize these properties of queuing system in instance-intensive business processes, we mainly apply input process as the incoming flow and service time as the execution time of scheduling tasks. We also employ a number of servers as the virtual machines in cloud environment. Buffer size represents the quantitative restriction of virtual machines which can be unlimited based on the dynamical property of cloud. Queue discipline is the basic scheduling rule by which a task will be selected. The most common rules are First In First Out (FIFO) and Random Serve by using queuing theory, and we aim at adjusting the number of virtual machines dynamically to control the length of waiting queue and response time. It is an effective way to save the cost of using cloud resources.

As shown in Figure 3, there can be many kinds of forms by using different properties in different queuing theories. Commonly used characters for "complex input process" and "service time distribution" in the shorthand notation are *D* (Deterministic), *M* (Markovian-Poisson for the arrival process or exponential for the service time distribution required by each task), *G* (General), GI (General and Independent), and Geom (Geometric) [33]. "Number of servers" can be a fixed number or a variable. "Buffer size" and "queue discipline" are designated as yellow color in Figure 3, which means that they can be omitted if they are unnecessary. "Buffer size" is not used if the waiting room is unlimited. "Queue discipline" is not used for the case of the First Come First Serve queue discipline.

5. Queuing Models for Business Workflow

5.1. Classical Models of Queuing Theory. There are some common forms of queuing systems. $G/G/1$ is the most general FIFO Single-Server Queue (SSQ) considered in queuing theory, where both the arrival and service processes are based on general distribution. G denotes a general distribution for both interarrival time and service time. One denotes that the model has a single server. The evolution of

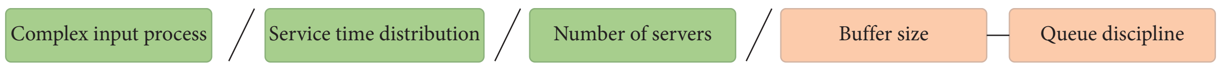

FIGURE 3: Fundamental form of queuing theory.

this queue can be described by the Lindley equation [34] which is a discrete-time stochastic process A_n where n is an integer value. Here, we set A_n to be the interarrival time between the n_{th} and $(n+1)_{\text{th}}$ tasks, B_n represents the service time of the n_{th} task. The process of execution can be used to describe the waiting time experienced by tasks in a queue or evolution of a queue length over time. Let W represent mean waiting time and we apply W_n to be the waiting time of the n_{th} task. So, the execution time of U_n can be described as

$$U_n = B_n - A_n. \quad (1)$$

Based on formula (1), we can figure out the waiting time of tasks in a recursion form:

$$W_{n+1} = \max\begin{cases} 0, \\ W_n + U_n, \end{cases} \text{s.t. } n \geq 1, \quad (2)$$

where $W_1 = 0$ represents that the first task does not need to wait. Subsequent tasks have to wait if they arrive at a time before the previous task has been served. Different interarrival and service time are considered to be independent so that sometimes the model is denoted as GI/GI/1 to emphasize the independent characteristic.

By considering the straightforward case of deterministic queues, we will discuss another form of queuing model where the interarrival and service time are nondeterministic. $M/M/1$ represents the queue length in a system which has a single server, where interarrival is determined by a Poisson arrival process, and task service time is based on an exponential distribution with infinite buffer. This model is the most elementary queuing system which is an attractive objective of study as closed-form expressions that can be obtained for many metrics of interest in this model. An extension of this model with more than one server is the $M/M/k$ queue.

The $M/M/1$ model is a stochastic process whose state space is the set $\{0, 1, 2, 3, \ldots\}$, where the value corresponds to the number of tasks in the system. Arrivals of tasks occur at rate λ according to the Poisson process and move the process from state i to state $i+1$. Service times of tasks have an exponential distribution with parameter $1/\mu$ in the $M/M/1$ queue, where μ means service rate. $M/M/1$ is a special case of $G/G/1$, so all the results which are applicable to $G/G/1$ are also applicable to $M/M/1$. Here, one important measure for performance of queuing system is the utilization, which is denoted as $\rho_{M/M/1}$. It is the proportion of time that a server is busy on average. The other probability of n tasks is denoted as p_n. Here,

$$\rho_{M/M/1} = \frac{\lambda}{\mu}. \quad (3)$$

By this utilization, we can get a balance equation which describes the probability flux associated with a Markov chain in and out of states or set of states in $M/M/1$ model. The balance equations of all tasks are shown as the following situations.

Situation 1:

$$\mu_1 P_1 = \lambda_0 P_0. \quad (4)$$

Situation 2:

$$\lambda_0 P_0 + \mu_2 P_2 = (\lambda_1 + \mu_1) P_1. \quad (5)$$

Situation n:

$$\lambda_{n-1} P_{n-1} + \mu_{n+1} P_{n+1} = (\lambda_n + \mu_n) P_n. \quad (6)$$

So, we can obtain the probability of P_0 in $M/M/1$ model by those balance equations.

$$P_0^{M/M/1} = \frac{1}{1 + \sum_{n=1}^{\infty} \prod_{i=0}^{n-1} (\lambda_i/\mu_{i+1})}. \quad (7)$$

$M/M/1$ is the simplest Markovian queue. A single machine is used to serve the first task at a time from the front of the queue according to FIFO discipline. When the service is finished, the task leaves the queue and the number of tasks in the system is decreased by one. Assuming that the $M/M/1$ queue-size process starts at state 0, it will stay in state 0 for a period of time that is exponentially distributed with parameter λ and then it moves to state 1. The buffer is of infinite size so there is no limit on the number of tasks. The model can be described as a continuous Markov chain with transition rate matrix on the state space.

$$Q^{M/M/1} = \begin{pmatrix} -\lambda & \lambda & & & \\ \mu & -(\mu+\lambda) & \lambda & & \\ & \mu & -(\mu+\lambda) & \lambda & \\ & & & & \ldots \end{pmatrix}. \quad (8)$$

Generally, state space transition diagrams are used to represent a system as a collection of states and activities associated with various relationships among the states. Queuing systems are modeled by continuous Markov chains that are often described by their state transition diagram which provides the complete information of their detailed balance equations. The state space transition diagram of $M/M/1$ is shown in Figure 4.

The diagram shows how the system moves from one state to another and the rate of movements between different

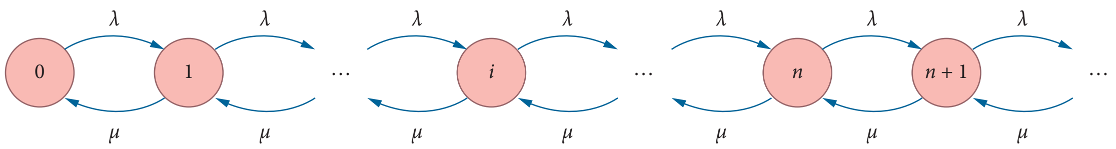

Figure 4: State space transition diagram of $M/M/1$.

states. The state space transition diagram has many applications related to the design and analysis of real-time and object-oriented systems.

5.2. M/G/k/l-P Model. As the business workflow example described in Section 3, there are tens of thousands of security transactions that need to be scheduled and handled in a fixed period of time. So, it is convenient to apply virtual machines as servers in cloud environment. In many cloud systems, server is paid for its usage time regardless whether it is busy or not. Normally, the time that transmission capacity is not used and this is time during which money is spent but no revenue is earned. Therefore, it is important to design systems that will maintain high utilization for cloud resources. Hence, we need to consider the waiting lengths of different models for various virtual machines, so we use l to represent the length of waiting queue. Finally, we add a service discipline in our model so it can be presented as $M/G/k/l$-P.

First, we will continuously discuss the first three basic components of $M/G/k/l$-P. Based on the above $M/M/1$ model in subsection 5.1, we can get the queuing model of $M/M/k$ when the applied number of servers is more than 1. The model can also be described as a continuous Markov chain with complex transition rate matrix on the state space.

$$Q^{M/M/k} = \begin{pmatrix} -\lambda & \lambda & & & & & \\ \mu & -(\mu+\lambda) & \lambda & & & & \\ & 2\mu & -(2\mu+\lambda) & \lambda & & & \\ & & & \vdots & & & \\ & & & k\mu & -(k\mu+\lambda) & \lambda & \\ & & & & k\mu & -(k\mu+\lambda) & \lambda \\ & & & & & & \vdots \end{pmatrix}. \tag{9}$$

The state space transition diagram of $M/M/k$ is shown in Figure 5.

There is a key difference between $M/M/1$ and $M/M/k$ since the system has m servers, which refers to the number of links between source and destination nodes. Let P represent the probabilities of n tasks in the system, S represent the number of times of entering state n, and L represent the number of times of leaving state n where S and L have the following relation $|S - L| \in \{0, 1\}$ when the system arrives at a steady state. Using the same calculation process with $M/M/1$, we can get the probability of p_0 in $M/M/k$ model. Here, the utilization of k servers for $M/M/k$ is denoted as $\rho_{M/M/k}$, where $\rho_{M/M/k} < 1$.

$$P_0^{M/M/k} = \left[\sum_{i=0}^{k-1} \frac{1}{i!}\left(\frac{\lambda}{\mu}\right)^i + \frac{1}{k!}\frac{1}{1-\rho_{M/M/k}}\left(\frac{\lambda}{\mu}\right)^k\right]^{-1}. \tag{10}$$

Then, the mean waiting time of $M/M/k$ model can be predicted so we can obtain the mean waiting time for all tasks of $W^{M/M/k}$ in formula (11). $\rho_{M/M/k} = \lambda/k\mu$; M means Markov that tasks arrive according to Poisson process.

$$W^{M/M/k} = \frac{(k\rho_{M/M/k})^k \rho_{M/M/k}}{k!\,(1-\rho_{M/M/k})^2 \lambda} P_0^{M/M/k}. \tag{11}$$

As the dynamic nature of business process, most of the tasks may be served in a period of time which is an arbitrary probability distribution. The distribution and mean number of busy servers are also insensitive to the shape of service time distribution. So, we apply $M/G/k$ model to schedule the business processes instead of $M/M/k$.

$M/G/k$ queue is a queuing model, where task arrivals have an exponential distribution with infinite buffer. G represents service time process which is arbitrary. k represents the number of virtual machines that we apply in cloud environment. The model is an extension of the $M/M/k$ queue, where service time must be generally distributed based on $M/G/1$ queue with a single server. Using the same calculation process with $M/M/k$, we can get the probability of p_0 in $M/G/k$ model. Here, the utilization of k servers for $M/G/k$ is denoted as $\rho_{M/G/k}$, where $\rho_{M/G/k} < k$.

$$P_0^{M/G/k} = \left[1 + \sum_{i=0}^{k-1} \frac{(k-1)!\,(k-\rho_{M/G/k})}{i!\,(\rho_{M/G/k})^{(k-1)}}\right]^{-1}. \tag{12}$$

Here, we make it clear that the insensitivity property does not extend to the arrival process. The distribution of busy servers and blocking probability is insensitive to the shape of the service time distribution. So, we concentrate on service time of different models. There are some elements that we need to use for illustrating the waiting time of $M/G/k$ such as expectation and variance. They are denoted as E and V in the following formula:

$$W^{M/G/k} = \frac{V + E^2}{2E\,(k-\rho_{M/G/k})} P_0^{M/G/k}. \tag{13}$$

The expectation of waiting time for $M/G/k$ model is also based on $M/M/k$. The main difference between them is the service time distribution. So, we compare both of the expectation for $M/M/k$ and $M/G/k$ in the following formula:

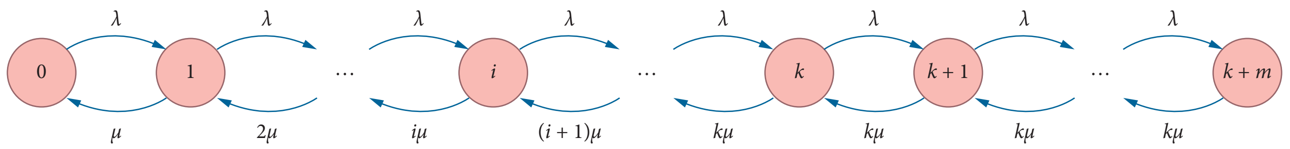

Figure 5: State space transition diagram of $M/M/k$.

$$E\left[W^{M/G/k}\right] = \frac{C^2+1}{2} E\left[W^{M/M/k}\right]. \quad (14)$$

Here, C^2 is a variable coefficient of service time distribution and C is less than 1. So, we can figure out that the value of $(C^2+1)/2$ is less than 1. It means that the expectation of $M/G/k$ will be less than $M/M/k$ based on formula (14).

Then, we add l to represent the length of waiting queue in our model by considering the waiting length of different models:

$$l = \frac{\lambda V + \lambda E^2}{2E\left(k - \rho_{M/G/k}\right)} P_0^{M/G/k}. \quad (15)$$

Finally, the most important factor we consider in this paper is the service discipline. This can significantly influence the Quality of Service (QoS) in instance-intensive business processes. There are various scheduling policies that can be used at queuing nodes. However, both the FIFO and Random Serve may result in quite long waiting time which can seriously influence the service time for most of tasks. In order to prove that all the business tasks can be scheduled in a constrained time period, we add the priority property which is represented by P in our queuing model. So, our model is described as $M/G/k/l$-P which means that tasks in instance-intensive business processes with high priority are served first.

Here, priority queues can be set into two types, nonpreemptive and preemptive. Nonpreemptive means a task in service cannot be interrupted and preemptive means a task in service can be interrupted by a higher priority task. So, the waiting queue in our $M/G/k/l$-P is dynamical based on different priorities of tasks and this can be called as dynamic process scheduling. The dynamic mechanism can satisfy some high QoS of users when they need to execute their tasks in a short time. It also can be used in the scenario that all tasks should be completed in a constrained time. As we discuss in Section 3 that security exchange is a typical instance-intensive business process, there are a large number of security exchange transactions need to be scheduled in a constrained time. Security company should prove that all the tasks are scheduled in the constrained time. If there is any error in the execution process which may cause execution congestion in a certain virtual machine. Then, the priority of tasks in waiting queue should be changed based on our dynamic process. So, it can adjust the execution process in real-time. This also can avoid the execution failure by the congestion, which may cause huge financial losses.

Based on the dynamic mechanism of $M/G/k/l$-P, if there is just one task in the system, the service rate of the system is μ_1, and only one virtual machine works in the scheduling process. It means the other virtual machines are all in idle state. If there are two tasks in the system, then the service rate of the system is $\sum_{j=1}^{2} \mu_j$. The service rate reaches the highest value when the task number reaches k, which means all the servers are put into operation. So, the state space transition diagram of Figure 6 is different with Figures 4 and 5.

We set four indicators like urgency degree, occupation degree, waiting time, and defrayment of tasks as x_1, x_2, x_3, and x_4. So, the priority of task can be set as

$$y_i = \frac{\max_{i=1}^{4}(x_i) - x_i}{\max_{i=1}^{4}(x_i) - \min_{i=1}^{4}(x_i)}. \quad (16)$$

Then, the proportion of the four indicators will be added based on formula (16). Finally, all the tasks are set into preemptive and nonpreemptive by our queuing theory. Based on the above discussion, a concrete process for $M/G/k/l$-P model is described in Algorithm 1, as the service discipline is applied in $M/G/k/l$-P that the tasks will be scheduled with different priorities. The normal time complexity is $O(\text{nm})$ in Algorithm 1 by nonpreemptive way, which is a very good result for scheduling. When some tasks have high priorities and need to preempt the resources of other tasks, the time complexity will be up to $O(\text{nm}^2)$ in the worst case.

6. Evaluation

6.1. Experimental Setting. In this section, we will present our simulation based on different parameters such as arrival time, waiting time, and response time for different tasks to test the effectiveness of our queuing model. The arrival time of tasks such as in security exchange will be different and follow a certain arrival rate. The clearing process in security exchange is a typically time-constrained and real-time system. All the tasks in security exchange must be completed based on a certain temporal expectation. However, there are thousands of security corporations which may have a great number of branches in security exchange market. After settling all the transaction processes during the trading day, the real fund settlement should be generated and cleared within a certain constrained time, which is most important for security exchange. The trading numbers of different security corporations may not be the same in different trading time of a certain day. So, the arrival rate of tasks will not be monotonous but in different range. It is appropriate to use our $M/G/k/l$-P queuing model because the task arrivals have an exponential distribution in our model, which

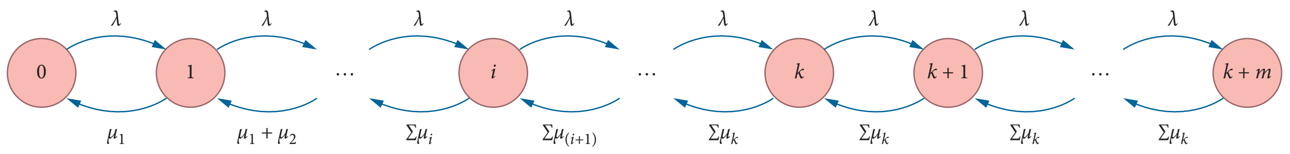

FIGURE 6: State space transition diagram of *M*/*G*/*k*/*l*-*P*.

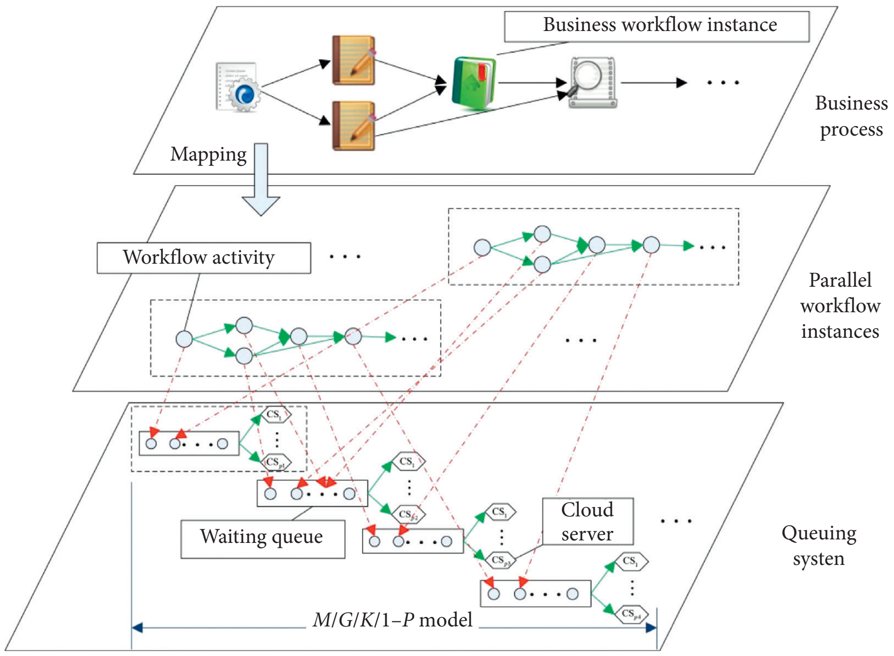

FIGURE 7: Queuing model for business workflows.

can represent the real service time process in security exchange. As depicted in Figure 7, massive business processes are mapped into workflow instances in a short period of time. The same type of cloud server is set in the waiting queue system for obtaining service.

Waiting time is also a most important parameter because security exchange is a real-time system. Each task needs to be handled as soon as it comes into the system to ensure the real-time requirement of the trading system. Waiting time is greatly influenced by the arrival time of tasks. However, waiting time can be totally different by various queuing systems. So, we mainly demonstrate our simulations to test the parameter of waiting time in this part. The other parameter we focus on is response time which depends on the tasks' arrival time, waiting time, and service time in the queuing system. The values of waiting time are expected to be small enough to satisfy the requirements of security exchange system. Then, these parameters will be carried out in different experiments and the contrast methods are First Come First Serve (FCFS) queue systems [35] and dynamic programming (DP) algorithm [36].

Our experiments are conducted based on various simulations to satisfy different requirements and conditions of security exchange systems. We compare various simulation results by different arrival rates, temporal expectation, waiting time, and response time. The simulation environment is based on Win 10 OS (32 GB memory/3.2 Hz CPU) and MATLAB 2015. The execution time in our simulation can be considered as standard time unit.

6.2. Experimental Results. Based on our queuing model *M*/*G*/*k*/*l*-*P*, the total waiting time is influenced by different arrival rate λ, service rate μ, and temporal expectation of all the tasks. If the temporal expectation is too small, then the number of servers needs to be more. So, we set a large temporal expectation first to avoid using large volume of servers as shown in Table 1. Here, the temporal expectation is set to 30 time units.

As shown in Table 1, the total waiting time is very high when both the arrival rate and service rate are very low. There is very short waiting time when the service rate is large enough. Generally, service rate should not be smaller than arrival rate so that waiting queue will not exceed the buffer size, which can avoid the overflow of waiting queue.

Then, some representative and comprehensive temporal expectations and service rates are applied to get convictive results in the following figures. The temporal expectation is set to 15, which is a relatively small number in Figure 8 to get a high demand result. In addition, the arrival rate is also set to a small number which can avoid a rather long waiting queue in the system.

As shown in Figure 8, five servers at least are required to deal with all the tasks because the total waiting time for

```
    Input: The arrival states of business workflow examples
    Output: The finished business workflow examples
(1)   **for** $i \longrightarrow 1$ **in** $k$ **do**
(2)     $l = \lambda V + \lambda E^2/2E(i-\rho)P_0$;
(3)     $y_i = \max_{i=1}^{4}(x_i) - x_i/\max_{i=1}^{4} - \min_{i=1}^{4}(x_i)$;
(4)     **if** $y_i > 1$ **do**
(5)       select the example of this priority to process
(6)       update $l$ and $y_i$;
(7)     **else**
(8)       **for** $u \longrightarrow 1$ **in** queue **do**
(9)         **if** $u$ **in** $l$ **then**
(10)          $u$ is selected to process
(11)          update $l$ and $y_i$;
(12)          **break**;
(13) $P = [1 + \sum_{i=0}^{k-1}(k-1)!(k-\rho)/i!\rho^{k-1}]^{-1}$;
(14) $W = V + E^2/2E(k-\rho)P$;
(15) update $P$ and $W$;
(16) **end**
```

Algorithm 1: $M/G/k/l$-P model.

Table 1: Total waiting time by $M/G/k/l$-P model with different arrival and service rates for all tasks.

λ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	83.3	—	—	—	—	—	—	—	—
0.2	33.3	41.7	—	—	—	—	—	—	—
0.3	21.4	23.4	27.8	—	—	—	—	—	—
0.4	15.9	16.7	18.2	20.8	—	—	—	—	—
0.5	12.6	13.0	13.8	14.9	16.9	—	—	—	—
0.6	10.5	10.7	11.1	11.8	12.6	13.9	—	—	—
0.7	9.0	9.1	9.4	9.8	10.3	10.9	11.9	—	—
0.8	7.8	7.9	7.9	8.3	8.7	9.1	9.7	10.4	—
0.9	7.1	7.1	7.1	7.3	7.5	7.8	8.2	8.6	9.3

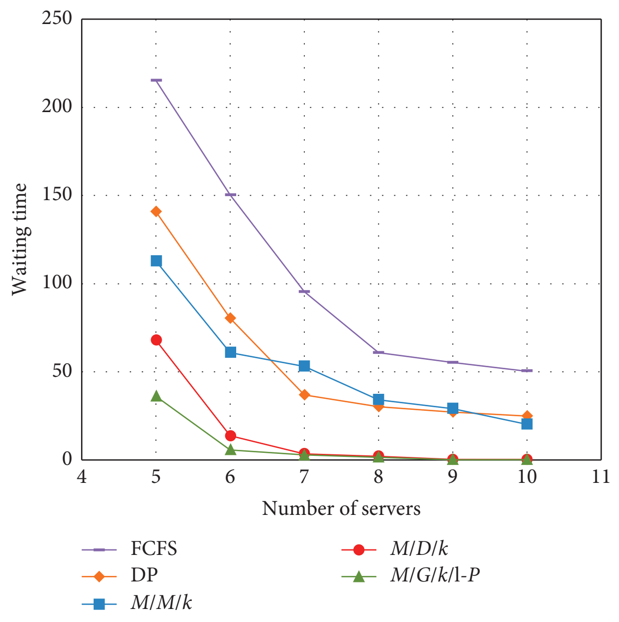

Figure 8: Total waiting time within the temporal expectation of 15 time units and arrival rate of 0.4 for all tasks.

all tasks will be very high if less than four servers are applied be it in FCFS, DP, $M/D/k$, $M/M/k$, or our $M/G/k/l$-P. However, the total waiting time decreases sharply when more servers are applied. Meanwhile, we can see that our $M/G/k/l$-P performs much better than the other methods regardless of how many servers are applied for scheduling.

Moreover, the total waiting time is also lower. However, if the arrival rate λ of tasks increases to 0.8 as shown in Figure 10, we can see that the total waiting time is higher than that in Figure 9. Specifically, when the number of servers is one or two, the waiting time in Figure 10 is much higher than that in Figure 9 because most of the tasks will swarm into the waiting queue at the beginning and the processing capacity cannot keep up with the demand of users when the arrival rate reaches to 0.8. So, most of the tasks need to wait for a long time when the applied servers are too few. This situation will cause a poor experience of QoS for users. In order to solve this problem, we can apply suitable number of servers according to the high arrival rate of 0.8 as shown in Figure 10. The waiting time is as much as that in Figure 9 by our $M/G/k/l$-P when applying more than two servers. Also, the waiting time obtained by $M/G/k/l$-P is much lower than the other methods.

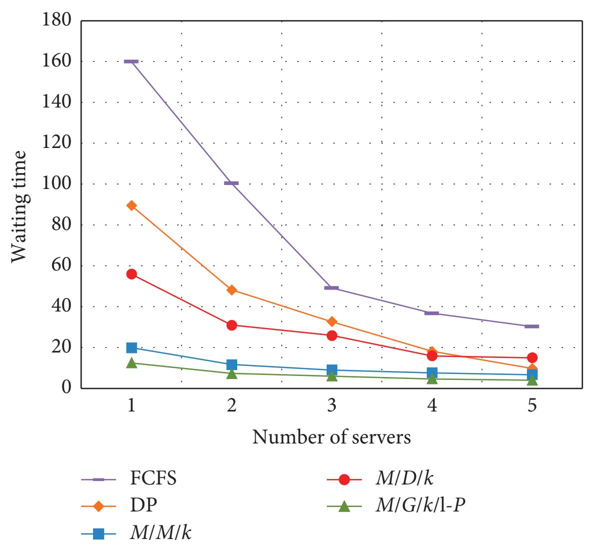

Figure 9: Total waiting time within the temporal expectation of 30 time units and arrival rate of 0.4 for all tasks.

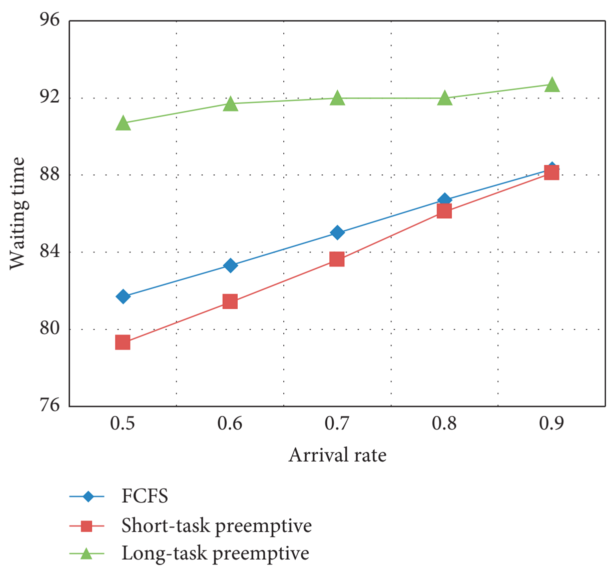

Figure 11: Total waiting time with different service disciplines.

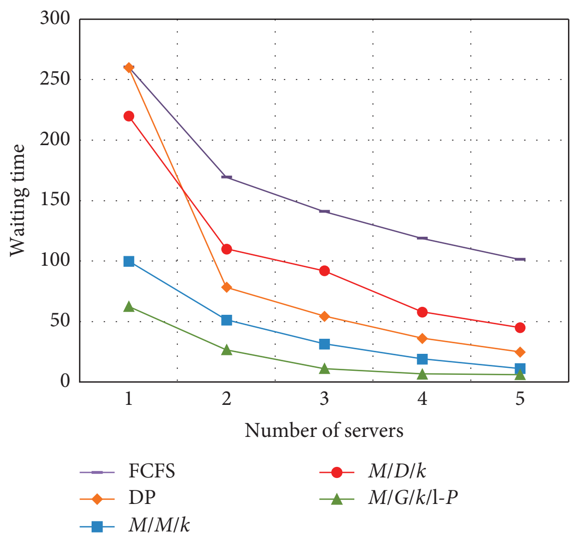

Figure 10: Total waiting time within the temporal expectation of 30 time units and arrival rate of 0.8 for all tasks.

Based on our model *M/G/k/l-P*, we can see that the waiting time is relatively small when applying a suitable number of servers, which influences the performance of the queuing model. Meanwhile, it can monitor the waiting time under different number of servers to dynamically adjust the number for queuing model. So, *M/G/k/l-P* can dynamically manage the number of virtual machines and waiting queue based on service time distribution to avoid much waste of cloud resources.

As we described in our queuing model *M/G/k/l-P*, service discipline is also an important factor that will significantly influence the experience of users. The default service discipline is FCFS and we set certain priorities to improve the QoS of users. As shown in Figure 11, the discipline of short-task preemptive performs better in waiting time than any other two disciplines when the arrival rate is lower than 0.9. Although the waiting time of long-task preemptive discipline is higher than the other models as shown in Figure 11, it can offer high priority for long task to avoid serving in queuing model at last. Hence, our queuing model *M/G/k/l-P* can meet the different demands of queuing question by considering corresponding service disciplines.

Figure 12 illustrates the changing process of total response time when the arrival rate increases. In this experiment, increasingly low values for the number of servers has a great impact on reducing the response time when the applied number of servers is less than four. Also, the response time can stabilize quickly when the number of servers increases. So, four servers are applied to see the variation trend when arrival rate increases by our queuing model *M/G/k/l-P* as shown in Figure 12. It demonstrates that the response time is highly influenced by arrival rate. However, the waiting time is smoothly influenced. Moreover, the waiting time has been reduced in large extent, which means our queuing model *M/G/k/l-P* actually improves the efficiency greatly.

Based on the above discussions, our model *M/G/m/k-P* can perform better than *M/D/k* and *M/M/k* models. We can know that the impact of arrival rate is smaller than time expectation in our model. All the experiments from Figures 8–11 indicate that *M/D/k*, *M/M/k*, and *M/G/k/l-P* get close performance while the applied numbers of servers are large enough because the differences between these models will be small and the redundant hardware resources will compensate for the lack of traditional models when applying more servers. However, these may cause huge waste to apply too much cloud servers. So, we proved that our model *M/G/k/l-P* can get the best results compared with others as shown from Figures 8–11 when we apply appropriate number of cloud servers. This feature

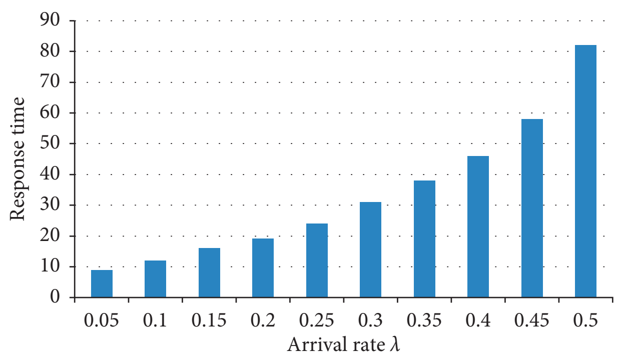

FIGURE 12: Total response time by *M/G/k/l-P* model.

ensures that our queuing model *M/G/k/l-P* can be well applied on the scenario of security exchange and shows great advantage compared with traditional models. Our queuing model *M/G/k/l-P* can provide correct and quick clearing process for large numbers of security transactions and save much execution cost for security exchange companies.

7. Conclusion and Future Work

The paper introduces a typical motivating instance-intensive business example in the stock market. It involves a large number of transactions which can be scheduled by queuing theory. Then, we put forward our novel model of queuing system based on queuing theory, which is *M/G/k/l-P* for business processes. Our model can handle the issue of allocating appropriate number of cloud resources dynamically to ensure that the number of cloud resources is sufficient and adequate. We also set different priorities for tasks based on the demands of users, so it can improve the experience of users. Various simulations in our paper prove that *M/G/k/l-P* performs very well for instance-intensive business process.

To the best of our knowledge, this is the first paper that schedules the instance-intensive business processes by queuing theory. The results presented in this paper promise a new research direction in the area of business workflows. However, there are at least two research topics that we need to address in the near future. First, the service rate should be larger than arrival rate so that the waiting queue would not exceed the size of buffer in our model *M/G/k/l-P*. Also, we can set a dynamical buffer in the future to relax this constraint. Second, the service discipline can be more flexible in the future so that the queuing model can perform even better.

Data Availability

The authors declare that materials described in the manuscript, including all relevant raw data, will not be available to any scientist.

Conflicts of Interest

The authors declare that they have no financial and personal relationships with other people or organizations that can inappropriately influence their work. There is no professional or other personal interest in any product and service that could influence the position presented in this manuscript.

Acknowledgments

The research was partly supported by the National Natural Science Foundation of China (61602005), MOE Youth Project of Humanities and Social Sciences of China (20YJCZH197), Natural Science Foundation of Anhui Province (1808085MF199), Natural Sciences Research Project in Universities of Anhui Province (KJ2018A0016), Putian Technology Planning Project (2019GP0011), and Starting funds of Putian University (2019020, 2019021). The authors are grateful to Professor Y. Yang, Australia Swinburne University of Technology, for editing the English language of the manuscript.

References

[1] X. Wang, L. T. Yang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 80–85, 2017.
[2] X. Xu, R. Mo, F. Dai, W. Lin et al., "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6172–6181, 2020.
[3] L. Qi, Y. Chen, Y. Yuan, S. Fu et al., "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web*, vol. 23, no. 2, pp. 1275–1297, 2019.
[4] H. Wu, W. J. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.
[5] X. Xu, Q. Liu, Y. Luo, K. Peng et al., "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.
[6] R. Xu, Y. Wang, H. Luo, F. Wang et al., "A sufficient and necessary temporal violation handling point selection strategy in cloud workflow," *Future Generation Computer Systems*, vol. 86, pp. 464–479, 2018.
[7] X. Liu, D. Wang, D. Yuan, F. Wang, and Y. Yang, "Workflow temporal verification for monitoring parallel business processes," *Journal of Software: Evolution and Process*, vol. 28, no. 4, pp. 286–302, 2016.
[8] X. Xu, Y. Xue, L. Qi, Y. Yuan, X. Zhang et al., "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, pp. 89–100, 2019.
[9] R. Xu, Y. Wang, W. Huang, D. Yuan et al., "Near-optimal dynamic priority scheduling strategy for instance-intensive business workflows in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 18, p. e4167, 2017.
[10] F. Amato and F. Moscato, "Exploiting cloud and workflow patterns for the analysis of composite cloud services," *Future Generation Computer Systems*, vol. 67, pp. 255–265, 2017.
[11] S. M. Nithya and V. R. Uthariaraj, "Identity-based public auditing scheme for cloud storage with strong key-exposure

resilience," *Security and Communication Networks*, vol. 2020, Article ID 4838497, 13 pages, 2020.

[12] A. Watanabe, K. Ishibashi, T. Toyono, K. Watanabe et al., "Workflow extraction for service operation using multiple unstructured trouble tickets," *IEICE Transactions on Information and Systems*, vol. E101.D, no. 4, pp. 1030–1041, 2018.

[13] X. Matsuo and H. Wu, "Spatio-temporal representation with deep neural recurrent network in MIMO CSI feedback," *IEEE Wireless Communications Letters*, vol. 9, no. 5, pp. 653–657, 2020.

[14] B. Cha, P. Sun, J. Kim et al., "International network performance and security testing based on distributed abyss storage cluster and draft of data lake framework," *Security and Communication Networks*, vol. 2018, Article ID 1746809, 14 pages, 2018.

[15] S. H. Bokhari, "A shortest tree algorithm for optimal assignments across space and time in a distributed processor system," *IEEE Transactions on Software Engineering*, vol. SE-7, no. 6, pp. 583–589, 1981.

[16] R. K. Chakrabortty, A. Abbasi, and M. J. Ryan, "A risk assessment framework for scheduling projects with resource and duration uncertainties," *IEEE Transactions on Engineering Management*, vol. 4, no. 10, pp. 1–15, 2019.

[17] Y. Zhao, R. N. Calheiros, G. Gange et al., "SLA-based profit optimization resource scheduling for big data analytics-as-a-service platforms in cloud computing environments," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 1–12, 2018.

[18] V. Nallur and R. Bahsoon, "A decentralized self-adaptation mechanism for service-based applications in the cloud," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 591–612, 2013.

[19] H. Chen, X. Zhu, G. Liu, and P. Witold, "Uncertainty-Aware online scheduling for real-time workflows in cloud service environment," *IEEE Transactions on Services Computing*, vol. 8, no. 7, pp. 98–110, 2019.

[20] Yu Jia, R. Buyya, and K. Ramamohanarao, "Workflow scheduling algorithms for grid computing," in *Metaheuristics for Scheduling in Distributed Computing Environments*, pp. 173–214, Springer, Berlin, Germany, 2008.

[21] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, "Imperfect information dynamic stackelberg game based resource allocation using hidden Markov for cloud computing," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 78–89, 2018.

[22] M. Guo, Q. Guan, W. Chen et al., "Delay-Optimal scheduling of VMs in a queueing cloud computing system with heterogeneous workloads," *IEEE Transactions on Services Computing*, vol. 10, no. 9, pp. 90–102, 2019.

[23] S. Singh and I. Chana, "Q-aware: Quality of service based cloud resource provisioning," *Computers & Electrical Engineering*, vol. 47, pp. 138–160, 2015.

[24] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[25] H. Luo, J. Liu, X. Liu, and Y. Yang, "Predicting temporal violations for parallel business cloud workflows," *Software: Practice and Experience*, vol. 48, no. 4, pp. 775–795, 2018.

[26] C. Wang, Z. Chen, K. Shang, and H. Wu, "Label-removed generative adversarial networks incorporating with K-Means," *Neurocomputing*, vol. 361, pp. 126–136, 2019.

[27] T. Matsubara, R. Akita, and K. Uehara, "Stock price prediction by deep neural generative model of news articles," *IEICE Transactions on Information and Systems*, vol. E101.D, no. 4, pp. 901–908, 2018.

[28] J. Liu, M. Sheng, Y. Xu, J. Li, and X. Jiang, "End-to-end delay modeling in buffer-limited MANETs: a general theoretical framework," *IEEE Transactions on Wireless Communications*, vol. 15, no. 1, pp. 498–511, 2016.

[29] Y. Xiang, V. Aggarwal, Y.-F. R. Chen, and T. Lan, "Differentiated latency in data center networks with erasure coded files through traffic engineering," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 495–508, 2019.

[30] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial IoT data scheduling based on hierarchical fog computing: a key for enabling smart factory," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4590–4602, 2018.

[31] S. Guo and M. Huang, "Simulation design of shop material supply based on queuing theory," in *Proceedings of the IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 258–263, Lisbon, Portugal, July 2018.

[32] H. Pourvaziri and H. Pierreval, "Dynamic facility layout problem based on open queuing network theory," *European Journal of Operational Research*, vol. 259, no. 2, pp. 538–553, 2017.

[33] C. Kim, A. Dudin, O. Dudina, and S. Dudin, "Tandem queueing system with infinite and finite intermediate buffers and generalized phase-type service time distribution," *European Journal of Operational Research*, vol. 235, no. 1, pp. 170–179, 2014.

[34] F. Ferreira, A. Pacheco, and H. Ribeiro, "Moments of losses during busy-periods of regular and nonpreemptive oscillating $$MX/G/1/n$$ M X/G/1/n systems," *Annals of Operations Research*, vol. 252, no. 1, pp. 191–211, 2017.

[35] M. Wang, W. Chen, and A. Ephremides, "Real-time reconstruction of A counting process through first-come-first-serve queue systems," *IEEE Transactions on Information Theory*, vol. 66, no. 7, pp. 4547–4562, 2020.

[36] Q. He, R. Zhou, X. Zhang et al., "Efficient keyword search for building service-based systems based on dynamic programming," in *Proceedings of the 15th International Conference on Service-Oriented Computing*, pp. 462–470, Malaga, Spain, November 2017.