A Queuing Theory Approach to Task Scheduling in Cloud Computing with Generalized Processor Sharing Queue Model and Heavy Traffic Approximation

Mohamed Ghazali, Abdelghani Ben Tahar

Abstract—**Cloud computing has transformed data storage, management, and processing by offering scalable and flexible resources via the internet. A key component of this technology is the efficient allocation and management of resources, particularly through task scheduling at the level of virtual machines (VMs). Task scheduling is critical for maximizing resource utilization and system performance in cloud environments. However, it presents significant challenges due to the dynamic and distributed nature of these environments. Effective task scheduling algorithms are necessary to balance load, minimize response time, and optimize resource usage, making it a crucial area for ongoing research and development in cloud computing.**

This paper addresses the challenge of task scheduling in cloud computing by employing an analytical approach based on queuing theory. We model the system using a generalized processor sharing (GPS) queue and evaluate its performance through heavy traffic approximation. This method allows us to derive performance metrics for queuing systems prone to congestion, considering general interarrival and service time distributions, thus providing a comprehensive analysis of scheduling efficiency.

Index Terms—**cloud computing, task scheduling, virtual machines, generalized processor sharing, heavy traffic approximation, performance evaluation, queue model.**

I. Introduction

In the rapidly evolving realm of information technology, the need for adaptable, scalable, and efficient computing solutions has become more pronounced than ever. Businesses and individuals are actively seeking innovative approaches to handle data, streamline operations, and promptly adapt to changing needs. In this dynamic context, traditional computing models often prove insufficient. It is within this context of technological advancement and the pursuit of increased efficiency that the concept of cloud computing emerges as a transformative influence. Cloud computing involves providing computing services over the internet, encompassing storage, processing power, and software. Instead of possessing and managing physical servers or computing infrastructure, individuals and businesses have the option to utilize these resources on a pay-as-you-go model through a cloud service provider. Key characteristics of cloud computing include on-demand self-service, where users autonomously manage resources for enhanced flexibility. The broad network access feature enables ubiquitous data and application access across various devices. Through resource pooling, providers dynamically allocate shared resources, optimizing efficiency. Rapid elasticity allows quick resource scaling to adapt to changing demand, ensuring optimal performance and cost efficiency. The measured service aspect involves metered resources, promoting a pay-as-you-go model for cost efficiency and transparency. Cloud computing provides diverse service models to meet user needs. Infrastructure as a service (IaaS) offers virtualized computing resources, such as virtual machines and storage. Platform as a service (PaaS) provides a platform with tools for application development, eliminating the need to manage underlying infrastructure. Software as a service (SaaS) delivers applications over the internet, removing the burden of installation, maintenance, and updates for users. In terms of elasticity and scalability, cloud services can scale both vertically (increasing resources within a single virtual machine) and horizontally (adding more virtual machines) to handle varying workloads. This flexibility ensures that applications can seamlessly grow or shrink in response to demand.

Within this dynamic context, queuing theory emerges as a mathematical approach for analyzing and modeling the flow of entities through a system of queues, playing a pivotal role in addressing challenges within the realm of cloud computing. Firstly, it can be used to create performance models for cloud systems. By analyzing the arrival rate of user requests, service times, and the number of servers available, performance metrics such as response time, throughput, and utilization can be predicted and optimized. Secondly, queuing theory helps in designing algorithms and policies for resource allocation and task scheduling. It aids in determining the optimal number of resources to allocate and the scheduling policies to minimize waiting times and maximize resource utilization. Additionally, it helps in designing effective load balancing mechanisms in cloud environments. By distributing incoming requests among different servers based on their current loads, it ensures that the system resources are utilized efficiently, reducing wait times and improving overall performance.

Cloud data centers comprise multiple physical servers, each hosting numerous virtual machines due to the principle of virtualization. Various types of jobs (web requests, database operations, storage, networking, etc.) are received by a specific physical server within the data center. Subsequently, a load balancer allocates these jobs to the virtual

Manuscript received March 28, 2024; revised September 6, 2024.

Mohamed Ghazali is a PhD student of Hassan First University of Settat, Faculty of Science and Technology, B.P. 577, Settat 26000, Morocco. (e-mail: m.ghazali@uhp.ac.ma).

Abdelghani Ben Tahar is a professor of Hassan First University of Settat, Faculty of Science and Technology, B.P. 577, Settat 26000, Morocco. (e-mail:abdelghani.bentahar@uhp.ac.ma).

machines. Given the diverse nature of arriving jobs, each with distinct types, the challenge lies in efficiently serving these jobs while ensuring fairness. Fairness, in this context, involves preventing low-priority jobs from being neglected. This objective is attainable through a service discipline known as generalized processor sharing (GPS). Under this service discipline, the server handles all jobs simultaneously, with the service shared among jobs in proportion to their priority. Consequently, high-priority jobs receive a greater share of the service compared to lower-priority ones. GPS can be used in many scenarious in the context of cloud computing. For instance, it finds relevance in modeling cloud hosting services where a provider offers web hosting to a varied clientele. Each client's websites are assigned weights based on hosting plans, subscription levels, or service requirements. Employing the GPS queue model allows the provider to prioritize high-paying clients or those with premium service agreements, allocating more significant processing resources to their websites. Simultaneously, clients with standard or basic plans receive a proportionally fair allocation, ensuring equitable resource sharing across the cloud infrastructure.

The remainder of the paper is organized as follows. Section II gives a brief overview of existing work in the literature. Section III provides an introduction to the GPS queuing system model within the framework of virtual machines. Section IV outlines the establishment of the heavy traffic approximation. Subsequently, Section V derives the performance metrics, and numerical results are presented in the same section.

II. Related work

In the landscape of cloud computing research, a diverse array of approaches has been investigated to tackle challenges and optimize different facets of system performance. In the pursuit of profit maximization, Cao et al. [1] utilize a single-server M/M/m queueing model, while Mei et al. [2] introduce a Double-Quality-Guaranteed (DQG) renting scheme employing an M/M/m+D queuing model. Numerous studies focus on performance evaluation, with Mas et al. [3] presenting a fog-computing modeling framework based on queuing theory, and Khazaei et al. [4] introducing an M/G/m/m+r queuing system model for assessing cloud server farm efficiency. Other works, such as those by Xia et al. [5], Chang et al. [6], Bai et al. [7], Liu et al. [8], Cheng et al. [9], and Guo et al. [10], delve into the intricacies of performance evaluation through queuing models, considering factors like migration-enabled clouds, active virtual machines, data center heterogeneity, and energy-saving algorithms. In the domain of system optimization, studies by Vilaplana et al. [11], Ali et al. [12], and El et al. [13] contribute novel insights using queuing theory to design cloud computing architectures, model horizontal elasticity, and efficiently scale fog computing systems for IoT devices, respectively. This comprehensive body of related work not only explores profit-driven motives but also emphasizes the significance of performance evaluation and system optimization across various cloud computing scenarios.

Most of these papers restrict themselves to the FCFS (First Come First Served) service discipline, employing exponential distributions for inter-arrival and service times. Only a limited number of papers in the existing literature incorporate a more general approach to interarrival and service times. In the majority of cases, these studies tend to confine themselves to considering only one of these parameters with a general distribution, typically focusing on service times. Notably, to our knowledge, no prior study has endeavored to model Virtual machines task scheduling using the Generalized Processor Sharing model.

III. System model

Figure 1 illustrates a GPS system within a cloud environment. Incoming jobs are categorized into distinct classes, and the server attends to the head of each flow according to the GPS scheduling discipline. Upon classification, requests are assigned to their respective categories, each associated with a weight factor that determines its priority level. Requests then enter their corresponding queues. The server processes requests from the front of each queue concurrently, allocating service capacity proportionally to the weight factors. This mechanism ensures fairness while prioritizing higher-priority classes. Figure 2 depicts our GPS queuing model. When all classes have non-empty queues, the total service rate is distributed according to the weight factors ω_k. If any queues are empty, the surplus service rate is reallocated proportionally among the remaining classes. Our queuing model consists of K classes of jobs, each arriving with a rate α_k. Assuming a server service capacity of $1/m$, the service rate for each class is given by:

$$\frac{\omega_k}{m\sum_{l=1}^{K}\omega_l}.$$

Let $u_k = (u_k(i), i \geq 1)$ represent the sequence of interarrival times for each class k, where $u_k(i)$ denotes the time between the $(i-1)$th and ith job arrivals. Similarly, let $v = (v(i), i \geq 1)$ be the sequence of service times, with $v(i)$ representing the service time of the ith job. We assume that $\{u_k\}_{k=1}^{K}$ and v are mutually independent sequences of independent and identically distributed (i.i.d.) random variables. For each class k, the sequence u_k has a mean of $1/\alpha_k$ and a variance of a_k. The service time sequence v has a mean of m and a variance of b. We focus on the heavy traffic regime where the traffic intensity ρ is close to 1:

$$\rho := m\sum_{k=1}^{K}\alpha_k \simeq 1.$$

Let $W(t)$ denote the workload at time t, defined as the sum of the remaining service times for all jobs present in the system at that time. Additionally, let $Z_k(t)$ represent the number of class k jobs in the system at time t.

IV. Heavy traffic approximation

Obtaining closed-form steady-state expressions for performance metrics in queuing systems with generally distributed interarrival and service times poses a significant challenge. To address this, we employ the heavy traffic approximation, also known as diffusion approximation, which provides accurate estimates, particularly when the traffic intensity approaches one [14]. This method analyzes a sequence of queuing networks indexed by positive real numbers $r \in \mathbb{R}_+^*$. The key step involves scaling these networks by accelerating time by a factor of r^2 and dividing space by a factor of r.

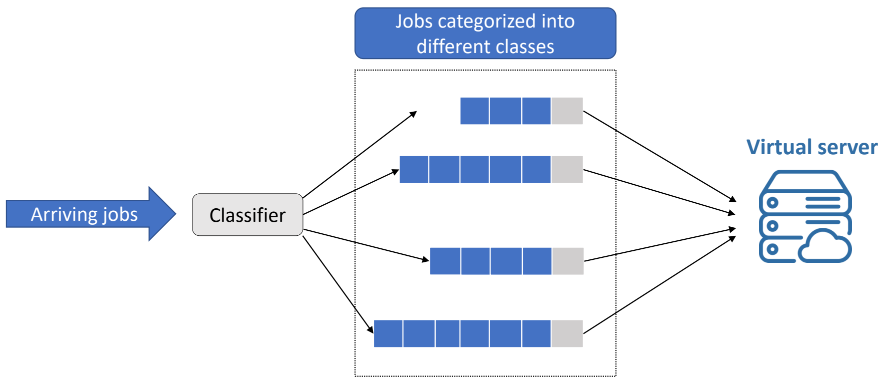

Fig. 1: Cloud system model.

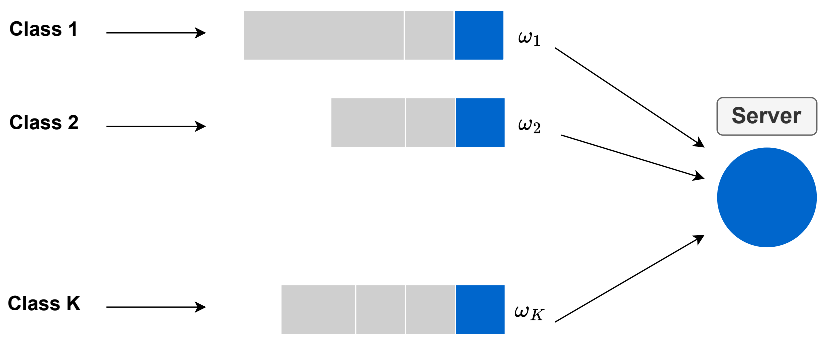

Fig. 2: GPS queuing model.

This scaling, denoted by a hat over the process, allows us to examine the limiting behavior as r approaches infinity. The resulting limit process is a reflected Brownian motion, which offers a significant advantage: it can be fully characterized using only the means and variances of the interarrival and service times at each queue. This property simplifies implementation and enables tractable performance analysis.

A. *Convergence of the diffusion scaled processes*

We consider a sequence of multiclass GPS queues indexed by $r \in (0, \infty)$. To represent the diffusion-scaled versions of the relevant parameters and processes, we introduce a superscript r. The diffusion-scaled queue length and workload processes are then given by:

$$\widehat{Z}_k^r(t) = \frac{Z_k^r(r^2 t)}{r}, \qquad \widehat{W}^r(t) = \frac{W^r(r^2 t)}{r}.$$

To establish the diffusion approximation, we first introduce the following convergence assumptions for each class k as r approaches infinity:

$$\alpha_k^r \longrightarrow \alpha_k, \quad a_k^r \longrightarrow a_k, \tag{1}$$
$$m_k^r \longrightarrow m_k, \quad b_k^r \longrightarrow b_k. \tag{2}$$

Furthermore, we assume the heavy traffic condition, which states that there exists a constant $\gamma < 0$ such that:

$$r(\rho^r - 1) \to \gamma. \tag{3}$$

Let $c_s^2 = b/m^2$ denote the squared coefficient of variation of the service distribution. For each class k, we define $c_{a,k}^2 = \alpha_k^2 a_k$ as the squared coefficient of variation of the arrival distribution.

Theorem 4.1: Under the assumptions of heavy traffic (1)-(3), the following diffusion-scaled processes converge in distribution:

$$\widehat{W}^r(t) \Rightarrow W^*(t), \tag{4}$$
$$\widehat{Z}_k^r(t) \Rightarrow Z_k^*(t) := \Delta_k W^*(t), \tag{5}$$

where for each class k:

$$\Delta_k = \frac{\alpha_k/\omega_k}{m \sum_{l=1}^K \alpha_l/\omega_l},$$

and W^* is a reflected Brownian motion with drift γ and variance Σ given by:

$$\Sigma = m^2 \left(\sum_{k=1}^K (c_s^2 + c_{a,k}^2)\alpha_k \right). \tag{6}$$

V. Performance evaluation.

A. *Performance metrics*

In this section, we leverage the result of Theorem 4.1 to derive some important performance metrics, including the average workload time, the average number of jobs, the average response time.

1) Average workload and average number of jobs: Since the traffic intensity is less than one, the queueing system reaches a steady-state distribution. In other words, the processes describing the system become stationary over time.

Specifically, by (4)-(5), we have:

$$\widehat{W}^r(\infty) \Rightarrow W^*(\infty),$$
$$\widehat{Z}_k^r(\infty) \Rightarrow Z_k^*(\infty) := \Delta_k W^*(\infty).$$

By the definition of convergence in distribution, this implies:

$$\mathbb{E}[\widehat{W}^r(\infty)] \to \mathbb{E}[W^*(\infty)]$$

and

$$\mathbb{E}[\widehat{Z}_k^r(\infty)] \to \mathbb{E}[Z_k^*(\infty)] = \Delta_k \mathbb{E}[W^*(\infty)].$$

From the definition of the diffusion workload process:

$$\mathbb{E}[W^r(\infty)] \simeq r\mathbb{E}[W^*(\infty)] \quad \text{for large } r.$$

Given that the workload limit process $W^*(\infty)$ is a reflected Brownian motion, we have:

$$P(W^*(\infty) > x) = \exp\left(\frac{2\gamma}{\Sigma}x\right),$$

which leads to:

$$\mathbb{E}[W^*(\infty)] = -\frac{\Sigma}{2\gamma}.$$

Using the heavy traffic condition (3), we can approximate $-\gamma/r$ by $(1-\rho^r)/\rho^r$. Therefore:

$$\mathbb{E}[W^r(\infty)] \simeq \frac{\Sigma}{2}\frac{\rho^r}{1-\rho^r}. \tag{7}$$

Furthermore, from the previous section, we know that for each class k:

$$\hat{Z}_k^r(t) \simeq \Delta_k \hat{W}^r(t) \quad \text{ for large } r,$$

which gives us:

$$\mathbb{E}[Z_k^r(\infty)] \simeq \Delta_k, \mathbb{E}[W^r(\infty)].$$

Substituting (7) and the definition of Δ_k into the above equation yields:

$$\mathbb{E}[Z_k^r(\infty)] \simeq \frac{\Sigma}{2}\frac{\rho^r}{1-\rho^r}\frac{\alpha_k/\omega_k}{m\sum_l \alpha_l/\omega_l}. \tag{8}$$

Finally, dropping the index r from (7) and (8), we obtain the following approximations for each class k:

$$\mathbb{E}[W] \simeq \frac{\Sigma}{2}\frac{\rho}{1-\rho} \quad \text{and} \quad \mathbb{E}[Z_k] \simeq \frac{\Sigma}{2}\frac{\rho}{1-\rho}\;\frac{\alpha_k/\omega_k}{m\sum_l \alpha_l/\omega_l}.$$

2) Average response time: Let Z_T represent the total number of jobs in the system. The average number of jobs can be expressed as:

$$\mathbb{E}[Z_T] = \sum_{k=1}^{K}\mathbb{E}[Z_k] \simeq \frac{\Sigma}{2}\frac{\rho}{1-\rho}\;\frac{1}{m}.$$

Consequently, the average response time within the system is given by:

$$\mathbb{E}[R] = \frac{\mathbb{E}[Z_T]}{\sum_{k=1}^{K}\alpha_k}.$$

It is important to note that both the average number of jobs and the average response time remain independent of the weight vector ω. This observation is consistent with the fact that these metrics assess the overall performance of the system and are not tailored to any specific class of jobs.

TABLE I: Performance measures for the case of exponential inter-arrivals and service times.

ω_1	ω_2	ω_3	$\mathbb{E}(W)$	$\mathbb{E}(Z_1)$	$\mathbb{E}(Z_2)$	$\mathbb{E}(Z_3)$
0.333	0.250	0.417	3.767	6.040	5.369	6.443
0.500	0.333	0.167	3.767	2.975	2.975	11.902
0.143	0.571	0.286	3.767	9.738	1.623	6.492

B. Numerical Results

In this section, we examine the operational dynamics of a virtual machine equipped with a single-core CPU within a data center. Jobs are initially directed to a specific physical machine and subsequently routed to a designated virtual machine by the load balancer. Our primary focus centers on the task scheduling at the virtual machine level. We presume the existence of a classifier that categorizes jobs into distinct classes. For the purpose of performance evaluation, we have established three categories, each associated with a respective weight factor. In our scenario, the first category experiences an influx of 1.5 jobs per millisecond, equivalent to a rate of 1500 jobs per second. For simplicity, we consider the millisecond (ms) as the unit of time. The arrival rates are denoted as $\alpha_1 = 1.5$, $\alpha_2 = 1$, and $\alpha_3 = 2$ for the three categories. The sequence of service times exhibits a mean of 4.74 milliseconds (ms), in other words the service rate is $m = 0.211$ jobs per millisecond (ms). It is important to note that these service times are inherently random and are computed by dividing the length of the jobs (measured in millions of instructions, MI) by the processing capacity of the virtual machine CPU (expressed in millions of instructions per second, MIPS). In this performance, we assume that all classes have the same coefficient of variation c_a^2 for for the inter-arrival times. Additionally, in this context, the traffic intensity ρ represent the VM utilization.

In Tables I and II, we present performance measures for different combinations of weights. Specifically, Table I illustrates this for exponential inter-arrival and service times, while Table II considers general inter-arrival and service times with varying coefficients of variation. We observe that in both tables, the average workload remains unchanged when we alter the weights. It is important to note that in Table II, the performance metrics remain consistent regardless of the distribution, as long as the coefficients of variation are the same. For example, two M/G/1 GPS systems, one with a log-normal distribution and the other with an Erlang distribution, will exhibit the same performance metrics if their coefficients of variation are identical.

As shown in Figures 3 and 4, the average workload and response time increase as VM utilization rises. Additionally, as the coefficients of variation increase, these metrics also become higher, which is evident from the figures. This is beacuse a high coefficient of variation for interarrival times means arrivals are really unpredictable, with some coming in quick succession and others after long gaps, leading to busy and quiet periods in the queue. For service times, it means some tasks are served quickly while others take much longer, making wait times unpredictable and staffing harder to manage, which can create bottlenecks.

In Figure 5, although class 3 has the highest weight (indi-

TABLE II: Performance measures for different coefficients of variation.

		ω_1	ω_2	ω_3	$\mathbb{E}(W)$	$\mathbb{E}(Z_1)$	$\mathbb{E}(Z_2)$	$\mathbb{E}(Z_3)$
$c_a^2 = 1/2$	$c_s^2 = 1/2$	0.333	0.250	0.417	1.883	3.020	2.685	3.222
		0.500	0.333	0.167	1.883	1.488	1.488	5.951
		0.143	0.571	0.286	1.883	4.869	0.811	3.246
$c_a^2 = 1/2$	$c_s^2 = 1/4$	0.333	0.250	0.417	1.413	2.265	2.013	2.416
		0.500	0.333	0.167	1.413	1.116	1.116	4.463
		0.143	0.571	0.286	1.413	3.652	0.609	2.434
$c_a^2 = 1/3$	$c_s^2 = 1/7$	0.333	0.250	0.417	0.897	1.438	1.278	1.534
		0.500	0.333	0.167	0.897	0.708	0.708	2.834
		0.143	0.571	0.286	0.897	2.319	0.386	1.546
$c_a^2 = 1/5$	$c_s^2 = 1/4$	0.333	0.250	0.417	0.848	1.359	1.208	1.450
		0.500	0.333	0.167	0.848	0.669	0.669	2.678
		0.143	0.571	0.286	0.848	2.191	0.365	1.461

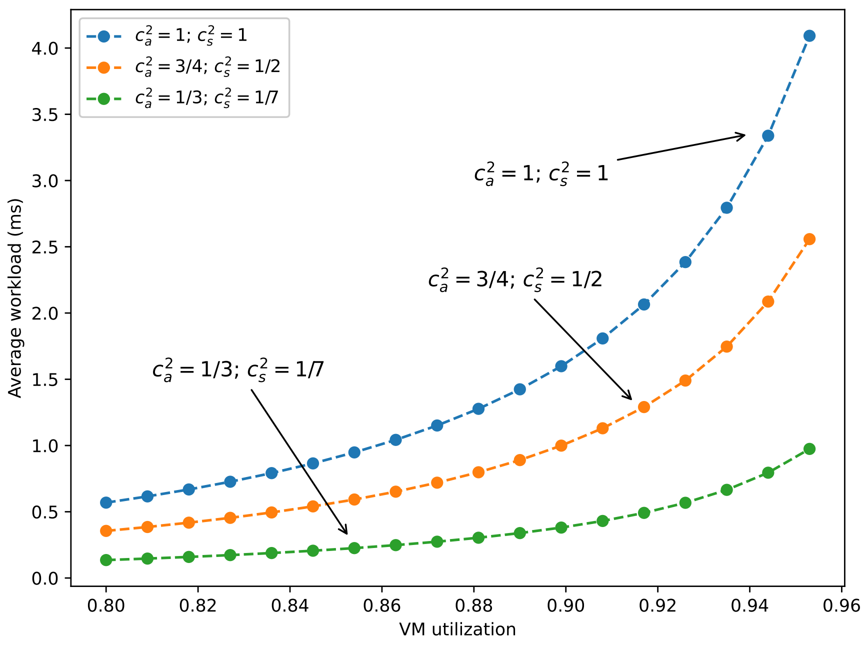

Fig. 3: The average workload as a function of VM utilization for different coefficients of variation.

cating it has the most service capacity among the classes), it also has the highest number of jobs. This occurs because its arrival rate is higher compared to the other classes. Note that the service rate is the same for all classes. If we assume the arrival rate is the same across all classes, then the class with the lowest weight (class 2) will have the highest number of jobs (and the highest response time), while class 3 will have the fewest jobs and the lowest response time. Class 1 falls in between the two.

In Figure 6, we compare GPS (Generalized Processor Sharing) and HLPPS (Head of the Line Proportional Processor Sharing). In HLPPS, service is allocated equally among the tasks at the head of each queue, regardless of their class. We observe that the average number of tasks varies between the two service disciplines based on the weight ω. The HLPPS represents the scenario where the weights are equal, specifically $\omega_1 = \omega_2 = \omega_3 = 1/3$. Consequently, for class 1, the average number of tasks is quite similar. However, for class 2, which has a weight of $1/4$, less than $1/3$, the average number of tasks is higher under the GPS service discipline because it receives less service allocation. For class 3, the opposite occurs, with a lower average number of tasks under GPS service due to its higher weight. HLPPS can be viewed as a special case of GPS with weights $w_k = 1/K$, where K is the number of classes.

VI. Conclusion

Our paper evaluates the task scheduling performance of virtual machines within the GPS service discipline. Despite focusing on a single virtual machine, our analysis extends to include a general distribution for both interarrival and service times, enriching the realism of our system dynamics representation. The performance metrics of the proposed queue system are systematically derived using a Heavy Traffic Approximation approach, offering valuable insights into its behavior under challenging conditions. Our study's

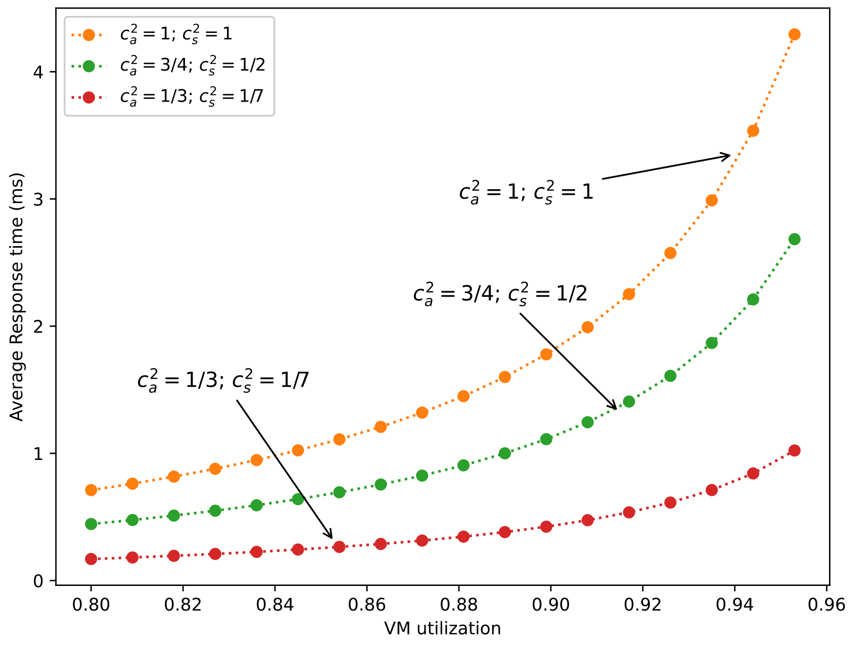

Fig. 4: The average response time as a function of VM utilization for different coefficients of variation.

focus on a single virtual machine, rather than the entire cloud system, is driven by the complexities of deriving closed-form performance results. This limitation leads us to examine the scenario of one VM with one core, while practical physical servers accommodate multiple VMs with multiple cores. In contrast, some literature explores more intricate systems but often resorts to simplifications, utilizing exponential distributions for interarrivals and service times and employing fixed probabilities for load balancing, sidestepping alternative strategies deemed intractable. Given these conditions, we can broaden our performance evaluation to include multiple VMs under the Generalized Processor Sharing service discipline.

Appendix A
Proof of Theorem 4.1

To establish our result, we leverage the framework presented in [15]. In that study, under heavy traffic conditions, the authors derived a heavy traffic approximation for various queuing disciplines, including the Generalized Processor Sharing (GPS) system referred to as GHLPPS in their paper. They consider a more general scenario where jobs can provide feedback, allowing them to reenter the queue and change their class. We define the vector $\Delta = (\Delta_1, \ldots, \Delta_k) \in \mathbb{R}^+$ as follows:

$$\Delta_k = \frac{\lambda_k/\omega_k}{m\sum_{l=1}^{K}\lambda_l/\omega_l} \quad \text{for each class } k.$$

In Theorem 3.2, they establish the heavy traffic limit for the processes $\hat{W}^r$ and $\hat{Z}^r$ and demonstrate the convergence of these processes in distribution:

$$(\hat{W}^r, \hat{Z}^r) \Rightarrow (W^*, Z^* = \Delta W^*) \quad \text{as } r \to \infty,$$

where W^* is a Reflected Brownian motion with drift $R\Sigma$ and variance $R^2\Sigma$. The constant R is defined in (3.2)-(3.3) in [15]; however, since there is no feedback, the matrix P is zero, resulting in $R = 1$.

As a remark, our queue model corresponds to the single station case, i.e., $J = 1$. Since there is no feedback, the matrix $Q = I$ and the matrix H is zero. The vector global arrival rate λ equals the the vector α. The matrix C equals the vector $e = (1, 1, \ldots, 1) \in \mathbb{R}_+^K$. The variance Γ is given by (3.8) and (3.12) in [15]. Considering the no feedback case, and the fact that $b_k = b$ for all classes k, and the matrix M is equal to mI, the variance becomes:

$$\Sigma = e\big(b \operatorname{diag}(\alpha_1, \ldots, \alpha_K) + m^2 \operatorname{diag}(\alpha^3 a_1, \ldots, \alpha_K^3 a_K)\big)e'.$$

Here e' denote the transpose of e. The variance equals

$$\Sigma = b\sum_{k=1}^{K}\alpha_k + m^2\sum_{k=1}^{K}\alpha_k^3 a_k.$$

Using the fact that $c_s^2 = b/m^2$ and $c_{a,k}^2 = \alpha_k^2 a_k$ for each k, we obtain

$$\Sigma = m^2\left(\sum_{k=1}^{K}(c_s^2 + c_{a,k}^2)\alpha_k\right),$$

which is exactly the variance in (6).

References

[1] J. Cao, K. Hwang, K. Li, and A. Y. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1087–1096, 2012.

[2] J. Mei, K. Li, A. Ouyang, and K. Li, "A profit maximization scheme with guaranteed quality of service in cloud computing," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3064–3078, 2015.

[3] L. Mas, J. Vilaplana, J. Mateo, and F. Solsona, "A queuing theory model for fog computing," *The Journal of Supercomputing*, vol. 78, no. 8, pp. 11138–11155, 2022.

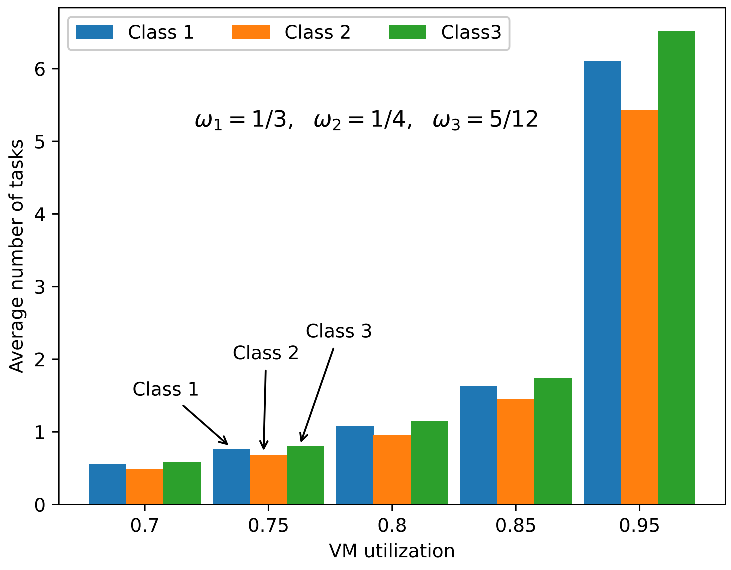

Fig. 5: The average number of tasks with exponential inter-arrival and service times.

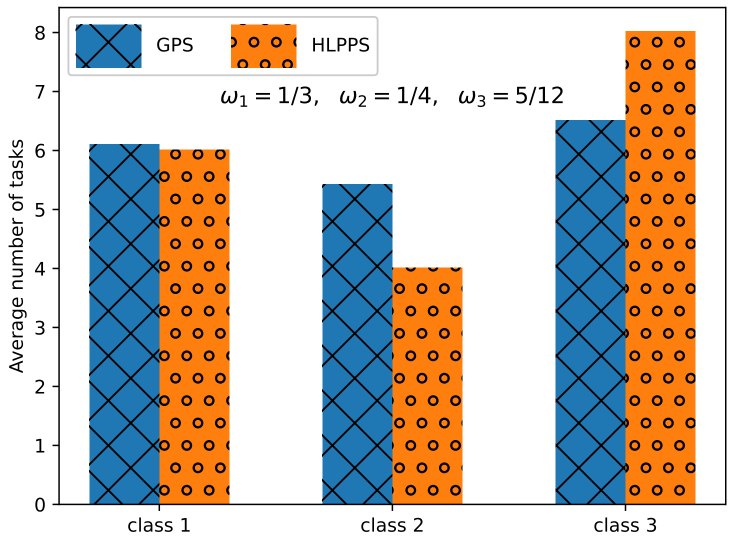

Fig. 6: Comparison of the average number of tasks between GPS and HLPPS with exponential inter-arrival and service times, and a VM utilization of 0.95.

[4] H. Khazaei, J. Misic, and V. B. Misic, “Performance analysis of cloud computing centers using M/G/m/m+ r queuing systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, pp. 936–943, 2011.

[5] Y. Xia, M. Zhou, X. Luo, S. Pang, and Q. Zhu, “Stochastic modeling and performance analysis of migration-enabled and error-prone clouds,” *IEEE Transactions on Industrial Informatics*, vol. 11, no. 2, pp. 495–504, 2015.

[6] X. Chang, B. Wang, J. K. Muppala, and J. Liu, “Modeling active virtual machines on IaaS clouds using an M/G/m/m+ k queue,” *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 408–420, 2014.

[7] W.-H. Bai, J.-Q. Xi, J.-X. Zhu, S.-W. Huang, *et al.*, “Performance analysis of heterogeneous data centers in cloud computing using a complex queuing model,” *Mathematical Problems in Engineering*, vol. 2015, 2015.

[8] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, “Multiobjective optimization for computation offloading in fog computing,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2017.

[9] C. Cheng, J. Li, and Y. Wang, “An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing,” *Tsinghua Science and Technology*, vol. 20, no. 1, pp. 28–39, 2015.

[10] L. Guo, T. Yan, S. Zhao, C. Jiang, *et al.*, “Dynamic performance optimization for cloud computing using M/M/m queueing system,” *Journal of Applied Mathematics*, vol. 2014, 2014.

[11] J. Vilaplana, F. Solsona, I. Teixidó, J. Mateo, F. Abella, and J. Rius, “A queuing theory model for cloud computing,” *The Journal of Supercomputing*, vol. 69, pp. 492–507, 2014.

[12] A. Ali-Eldin, J. Tordsson, and E. Elmroth, “An adaptive hybrid elasticity controller for cloud infrastructures,” in *2012 IEEE Network Operations and Management Symposium*, pp. 204–212, IEEE, 2012.

[13] S. El Kafhali and K. Salah, “Efficient and dynamic scaling of fog nodes for IoT devices,” *The Journal of Supercomputing*, vol. 73, pp. 5261–

5284, 2017.

[14] M. Miyazawa, “Diffusion approximation for stationary analysis of queues and their networks: a review,” *Journal of the Operations Research Society of Japan*, vol. 58, no. 1, pp. 104–148, 2015.

[15] M. Bramson and J. G. Dai, “Heavy traffic limits for some queueing networks,” *Annals of Applied Probability*, pp. 49–90, 2001.