WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Different Modeling Techniques for IaaS Cloud

Surabhi Sachdeva*, Shalini Gambhir

Department of Computer Science Engineering, K. R Mangalam University
Gurugram, India
surabhisachdeva@gmail.com

A B S T R A C T

Mathematical analysis and cloud service performance evaluation has always been a difficult and time-consuming task. When the input parameters are changed, the analytical models-based evaluation aids in determining the system's performance. Most queuing theory models either assume a Markovian distribution to simplify the mathematical process or propose a complex mathematical approach to evaluating the parameters of interest. This article summerise the different system model techniques used for IssS cloud computing. One of the main goals is to demonstrate the importance of the tradeoff between the number of servers, service rate, and system throughput. The findings of the study concludes that GEM, which decomposes complex network scenarios into simpler and solvable open queuing networks, In comparison to other analytical models.

Keywords: Blocking Probability, Buffer, General Expansion Method, Network Throughput.

1. Introduction

Cloud-based infrastructure-as-a-service (IaaS) is growing in popularity as a diverse and active branch of commercial ICT services. Users of IaaS clouds can provision "processing, storage, networks, and other fundamental resources" on-demand, that is, when and for as long as they need them, and only pay for what they use. Commercial IaaS clouds, such as Amazon's EC2, have grown in popularity over the last five years, with users ranging from small and medium businesses to scientific HPC users(Ardagna, Casale, Ciavotta, Pérez, & Wang, 2014; Zhou & He, 2014). However, increased cloud adoption, and possibly even pricing models, are contingent on (potential) cloud users' ability to benchmark and compare commercial cloud services. From the user's perspective, we investigate the IaaS cloud-specific elements of benchmarking in this chapter. Good performance is an important feature of IaaS clouds, which must be guaranteed on-demand and maintained over a long period of time. However, as we've seen with several other new technologies in their early stages, most notably grid computing in the 1990s, IaaS clouds are likely to go through a period of changing performance management practises. We expect the branch of performance management that focuses on measuring performance to evolve from traditional practises in order to meet the needs of cloud operators and customers(Ardagna et al., 2014; Zhou & He, 2014).
Benchmarking is a tried-and-true method of ensuring that a system's performance meets its specifications. Consumers can easily compare products and put pressure on providers to use best-practices and possibly lower costs when benchmarking results are published, for example through mixed consumer-provider organisations like SPEC and TPC. Cloud computing is currently used in a variety of applications, including hosting applications, media, games, and websites, E-commerce, On-Demand Workforce and CRM, high-performance computing, search, and raw resources for various purposes. Commercial clouds must meet their own (de facto) performance standards for each application area, and some have even developed benchmarks (e.g., BioBench for Bioinformatics and RUBiS for online business). We believe that three major trends emerging from the last decade of grid and large-scale computing can be used to predict the characteristics of current and near-future workloads in IaaS clouds. First, individual jobs are increasingly being divided into smaller compute or data-intensive tasks (many tasks; there are almost no tightly coupled parallel jobs left. Second, the duration of individual tasks is decreasing with each passing year; only a few tasks last longer than an hour, and the majority take only a few minutes to complete. Third, compute-intensive jobs are divided into BoTs or DAG-based workflows, whereas data-intensive jobs can use a variety of programming models, including MapReduce and general dataflow.

Cloud benchmarking is not a simple extension of traditional benchmarking techniques. Several large-scale computing environments with cloud-like characteristics have existed in the past. CERN and the IBM T.J. Watson Research Center, for example, had large numbers of mainframes (using virtualization through the Virtual Machine operating system!) and used multi-tenancy across their departments decades ago. Similarly, some vendors had

large-scale installations that customers could use for a fee via Remote Job Entry services. Benchmarking and capacity planning were carried out in these environments in close collaboration between owners and customers. Customers who want to benchmark their potential computing environments can simply use credit card access to deploy and benchmark their applications in the cloud: clouds offer not only elasticity on demand, but also (resources for) capacity planning and benchmarking on demand. Customers will have to gain sufficient trust in the performance, elasticity, stability, and resilience of clouds through benchmarking in order to rely on them for the operation of their businesses. In fact, cloud customers may want to benchmark both before and after migrating to the cloud to assess the ongoing performance of their applications in the cloud. As a result, the ability of cloud benchmarks to allow users to gain trust without requiring lengthy setups or expensive operation is critical(Farid, Latip, Hussin, & Abdul Hamid, 2020; Strumberger, Bacanin, Tuba, & Tuba, 2019).Despite the numerous approaches to benchmarking and performance evaluation of various systems that have emerged in recent years, there is no unified view of the major challenges that researchers and practitioners in the field of benchmarking face. This chapter aims to provide that unified view, and it should be useful in system procurement and performance management as a result. The unified view draws on previous work on benchmarking middleware, databases, grid and parallel-system scheduler performance evaluation, and benchmarking systems in general (Adhikari & Srirama, 2019; Asadi, Azgomi, & Entezari-Maleki, 2019).

A generic architecture for IaaS cloud benchmarking is included in the unified view. The architecture has been designed to be familiar to existing practitioners while also providing new, cloud-specific functionality. Current IaaS cloud operators, for example, lease resources to their customers but leave the selection of resource types and lease/release moments to the customers; because such selection can have a significant impact on the performance of the system built to use the leased resources, the generic benchmarking architecture must include policies for resource provisioning and allocation.

The unified view introduced in this chapter focuses on ten important methodological, system, workload, and metrics-related issues in addition to traditional benchmarking elements and the generic architecture. How should cloud bursting systems, which lease resources to supplement a customer's own resources, be benchmarked, for example? What are some realistic workload models for IaaS clouds? How can IaaS clouds that share resources among multiple customers be judged on their ability to isolate user environments and thus prevent performance variability (Carvalho, Menascé, & Brasileiro, 2017; Ghosh, Longo, Frattini, Russo, & Trivedi, 2014).

1.1. Benchmarking Computer Systems: A Primer

Benchmarking is the process of evaluating a computer system's performance and other non-functional characteristics in order to compare it to other systems or industry-accepted standards. Benchmarking has traditionally been used to aid in the informed procurement of computer systems by allowing system vendors and third parties to publish verifiable results. The requirement of using SQL for this benchmark has resulted in widespread acceptance of the ANSI SQL92; additionally, the complexity of the majority of the queries has resulted in numerous improvements in aggregate function design and support. As a result of this benchmark, the geometric mean has become widely used for aggregating normalised results. The tuning of the DAS multi-cluster system has benefited from the benchmarking activity of some of the authors of this chapter, which began in the mid-2000s; at the time, our distributed computing benchmarks revealed a number of (fixable) issues with the in-operation system.

The lack of expertise of potential users is one of the most significant impediments to the adoption of a new technology. Personnel shortages in computer science are a major source of concern for the European Union and the United States. Benchmarks can represent best-practices and thus be valuable training material due to their open-source nature and representation of industry-accepted standards.

On Benchmarking Alternatives For the purposes described earlier in this section, a variety of alternative methods have been used, including empirical performance evaluation, simulation, and even mathematical analysis. Benchmarking, in our opinion, is an empirical evaluation of performance that adheres to a set of accepted procedures and best practises. As a result, using empirical performance evaluation can be beneficial, but it may lack the representativeness of a (de facto) standard benchmark. When the behaviour of the system is well understood and for long-running evaluations that would be impractical otherwise, we see a role for (statistical) simulation and mathematical analysis. Simulating new technology, such as cloud computing, on the other hand, necessitates careful (and time-consuming) validation of assumptions and models(Entezari-Maleki, Sousa, & Movaghar, 2017; Ghosh, Longo, Naik, & Trivedi, 2013).

1.2 Benchmarking Components

We review the main elements of a benchmarking process here, based on canonical texts. The main characteristics of a benchmark—relevance, portability, scalability, and simplicity—have been extensively discussed in related literature, such as (Brebner, 2012; Jhawar & Piuri, 2012). The system being evaluated is known as the System Under Test (SUT). A white box system reveals its entire operation, whereas a black box system hides operational details and is evaluated solely on the basis of its outputs.

The workload refers to the operational load placed on the SUT. Simple micro benchmark are simplified or reduced size codes designed to stress potential system bottlenecks, based on the empirical observation that "20 percent of the code consumes 80 percent of the resources." The results of micro benchmarks can be combined with application profiles to provide credible performance predictions for any platform using the methodology(Iosup, Prodan, & Epema, 2014b; Liu, Chang, Han, Trivedi, & Rodríguez, 2018). As a result of system improvements that make microbenchmarks run quickly but have no effect on the performance of much larger codes, synthetic and even real-world (complex) applications are now used for benchmarking. Simple workloads comprised of a single application and a (realistic) job arrival process have been used for benchmarking in distributed and large-scale systems such as IaaS clouds [33]. Simple workloads comprised of a single application and a (realistic) job arrival process represent better the typical system load

and have been used for benchmarking in distributed and large-scale systems such as IaaS clouds. Complex workloads, or the sum of multiple users' simple workloads, possibly with different applications and job characteristics, have begun to be used in the evaluation of distributed systems(Amato, Moscato, Moscato, & Colace, 2018; Ataie et al., 2017; Cloud, Moreno-vozmediano, Montero, & Llorente, 2012; Comput et al., 2012; Iosup, Prodan, & Epema, 2014a; Manvi & Krishna, 2013; McDole, Abdelsalam, Gupta, & Mittal, 2020), and we believe they will play an important role in benchmarking.

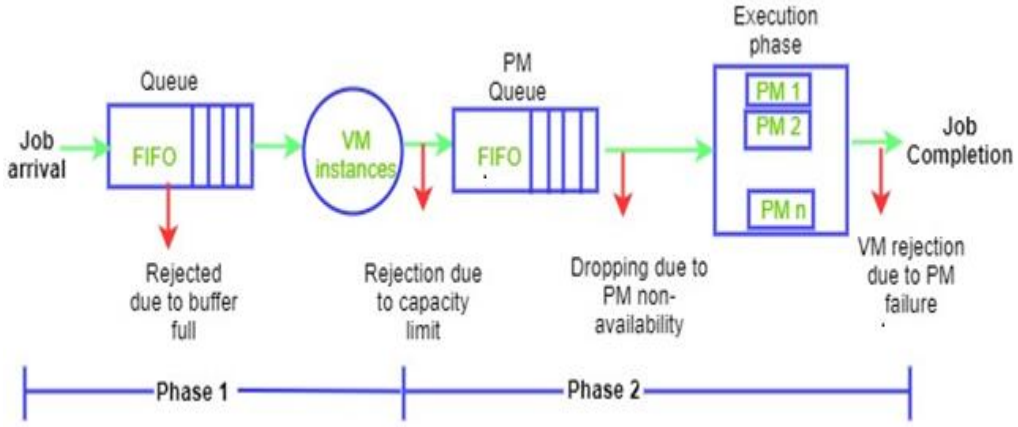

Fig.1. Mathematical System Model for Cloud Data Center Jobs Processing [11]

2. System Model

The two-stage tandem network is depicted in Figure 1 as a cloud process. The first stage is known as the CMU Handling phase/Resource Provisioning module and Resource Assigning module in the literature. Job Execution phase/Task Scheduling and VM Mapping/VM Provisioning Module are the names for the model's second stage.

Queues sit in front of the first and second units, handling user requests and virtual jobs assigned to physical machines, respectively. To make mathematical operations easier, the following assumptions are made: I Both queues are organised according to the First Come First Serve (FCFS) principle. (ii) A single job can be deployed on a single physical machine, which can only handle one job at a time.

Users submit a job request to the first stage. The jobs that are received are split up into different virtualized instances. The output of the first stage is used as the input for the second stage. In the literature, the second-stage service rate is usually assumed to be either exponential or linear. However, the machine's service time cannot always be considered an exponential function because it does not reveal the machine's true nature. The proposed solutions for evaluating performance using generalised service time are mathematically challenging.

Using the generalised expansion method, the current paper simplifies performance evaluation for generalised service distribution function problems (GEM). The proposed method can be used to analyse the performance of networks that are arbitrarily configured, finite queuing, or acyclic. Network reconfiguration, parameter estimation, and feedback elimination are the three stages of the process.

3. Analyzing Different Queue Scenario's in Cloud

3.1 Open Queue network with Single Resource Provisioning Server and Single VM Provisioning Server

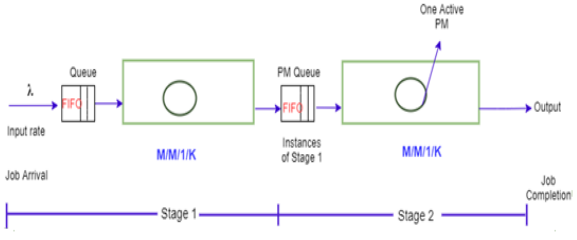

Fig.2. Finite Capacity Single Resource Provision module and Single VM Provisioning Server

In our first model, model 1A, both the resource provisioning module (node1) and VM provisioning stage (node2) are assumed to implement M/M/1/K queues. We can apply either Jackson's theorem or GEM approximation to get desired parameters.

Let λ represent the mean inter-arrival time at node1 and K denote the buffer capacity for both node1 and node2. The utilization factor, also referred to as traffic intensity, for node 1 and node 2 can be expressed as:

$$\rho_0 = \lambda / \mu_0 \quad (1)$$

$$\rho_1 = \lambda_1 / \mu_1 \quad (2)$$

where μ_0, μ_1, and λ_1 represent the mean service time of node 1, mean service time of node 2 and mean inter-arrival time at the node 2, respectively. The probability mass function, P(K0) and P(K1), represents the probability that the buffer of node1 and node2 is full. Mathematically it can be expressed as:

$$P(K_0) = \frac{(1-\rho_0)\rho_0^{K}}{\left(1-(\rho_0)^{K+1}\right)} \quad (3)$$

$$P(K_1) = \frac{(1-\rho_1)\rho_1^{K}}{\left(1-(\rho_0)^{K+1}\right)} \quad (4)$$

The Jackson theorem states that given the distribution of job inter-arrival times and service times at all the nodes be exponential and the scheduling algorithm be FCFS, the joint probability of the system denoted by p(K0, K1) can be expressed as the product of the steady-state probability mass function of K0 and K1 jobs at the node1 and node2, respectively. The term P(K0, K1) represents the number of the jobs that are waiting in the system, including node 1 and node 2.

$$P(K_0, K_1) = P(K_0) * P(K_1) \quad (5)$$

$$P(K_0, K_1) = \frac{(1-\rho_0)\rho_0^{K}}{\left(1-(\rho_0)^{K+1}\right)} * \frac{(1-\rho_1)\rho_1^{K}}{\left(1-(\rho_0)^{K+1}\right)} \quad (6)$$

By using (6) and (7), the throughput of the M/M/1/K system is:

$$T_{put} = \lambda * \left(1 - \left(\frac{(1-\rho_0)\rho_0^{K}}{\left(1-(\rho_0)^{K+1}\right)} * \frac{(1-\rho_1)\rho_1^{K}}{\left(1-(\rho_0)^{K+1}\right)}\right)\right) \quad (7)$$

With the help of Figure 3, we illustrate the working of the GEM algorithm, which is based on the principle of blocking after service.

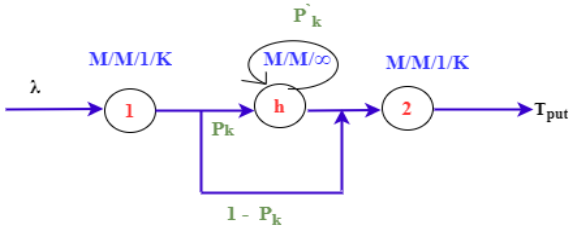

Fig.3. Expansion of the M/M/1/K as per Generalized Expansion Method

Network re-configuration. Insertion of an artificial finite capacity node between node1 and node2 in the network. The holding node 'h' preceded node2 and modeled as an M/M/∞ queue. The service rate for the artificial node is assumed to be μ_h. The jobs serviced out from node 1 proceed for node 2 and buffered in the VM provisioning module depending upon the number of jobs serviced. Node2 can either be in a saturated state or an unsaturated state. Once in the saturated state or blocked state, it cannot accommodate more jobs; that is, its buffer is full Otherwise, the node is said to be in the unblocked state or unsaturated state. Let the probability of blocking the job be represented by P_k. The expression 1-P_k represents the unblocking probability of node2. Whenever node2 is in the blocked state, incoming jobs are rerouted to node h and suffer an extra delay. The parameter $P_k^{'}$ is the probability that the job continues to stay in node h until accommodated in the buffer of node2. It is essential to mention here that the infinite number of servers represents the fact that the blocked job is suffering the delay without queuing.

i. Parameter Estimation. IIt is essential to estimate the value $P_K, P^{'}_K, \mu_h$ by utilizing the known results for M/M/1/K. The value of P_K is expressed as:

$$P_K = P\left(K\,jobs\,in\,the\,queue\,or\,service\right) = \frac{\left(1-\rho_1\right)\rho^K}{1-\left(\rho_1\right)^{K+1}} \tag{8}$$

The value of $P_k^{'}$ can be approximated using the diffusion technique given by (Labetoulle & Pujolle, 1980)

$$P_K^{'} = \left(\frac{\mu_1+\mu_h}{\mu_h} - \frac{\lambda\left[\left(r_2^K - r_1^K\right) - \left(r_2^{K-1} - r_1^{K-1}\right)\right]}{\mu_h\left[\left(r_2^{K+1} - r_1^{K+1}\right) - \left(r_2^K - r_1^K\right)\right]} \right) \tag{9}$$

where r_1 and r_2 are the roots of the below-mentioned quadratic equation.

$$\lambda - \left(\lambda + \mu_h + \mu_1\right)x + \mu_h x^2 = 0 \tag{10}$$

$$r_1 = \frac{\left(\lambda + 2\mu_h\right) - z^{1/2}}{2\mu_h} \tag{11}$$

$$r_2 = \frac{\left(\lambda + 2\mu_h\right) + z^{1/2}}{2\mu_h} \tag{12}$$

The dummy parameter 'z' is mathematically expressed as:

$$z = \left(\lambda + 2\mu_h\right)^2 - 4\lambda\mu_h \tag{13}$$

The variable λ is determined using equation 15:

$$\lambda = \lambda_1 - \lambda_h\left(1 - P^{'}_K\right) \tag{14}$$

where λ_1 and λ_h are the actual arrival rate to the node2 and the actual arrival rate to artificial node h.

Since the service time distribution in node 2 is exponential, the mean service time distribution of the artificial node is equivalent to the service time distribution of node2, that is:

$$\mu_h = \mu_1 \tag{15}$$

ii. Feedback elimination. The feedback loop creates strong dependencies in arrival processes. To avoid such occurrences, a reconfiguration of the holding node is performed. To eliminate the feedback arc, the service time at the artificial node needs to be recomputed as expressed as $\mu_h^{'}$

$$\mu^{'}_h = \left(1 - P^{'}_K\right)\mu_h \tag{16}$$

The buffer state of the node2 being dictates the mean service time of node 1. Analytically, the new mean service time of node 1 can be expressed as:

$$\tilde{\mu_0}^{-1} = \mu_0^{-1} + P_K\left(\mu^{'}_h\right)^{-1} \tag{17}$$

All the parameters of interest can be determined by simultaneously solving the set of equations (8) to (15). The system's response time is the time for which a user waits for the job to be processed. It can be determined as:

$$RT = \frac{1}{\lambda_{eff}} * \left(P\left(K_0, K_1\right)\right) \tag{18}$$

Utilizing (6) and (18), the response time of the M/M/1/K system is:

$$RT = \frac{1}{\lambda_{eff}} * \left(\frac{\left(1-\rho_0\right)\rho_0^K}{\left(1-\left(\rho_0\right)^{K+1}\right)} * \frac{\left(1-\rho_1\right)\rho_1^K}{\left(1-\left(\rho_0\right)^{K+1}\right)} \right) \tag{19}$$

In our second model, model 1B, the node1 and node2 modules follow M/G/1/K queuing model. Jackson's theorem is applicable only for exponential distribution service rates. In the present case the same cannot be applied since the service time in both the nodes have general distribution model. For the non-exponential open finite queuing networks GEM provides a set of equations that should be solved simultaneously as illustrated in model 1A.

The blocking probability for a node having generalized service time distribution can be determined using (MacGregor Smith, 2011):

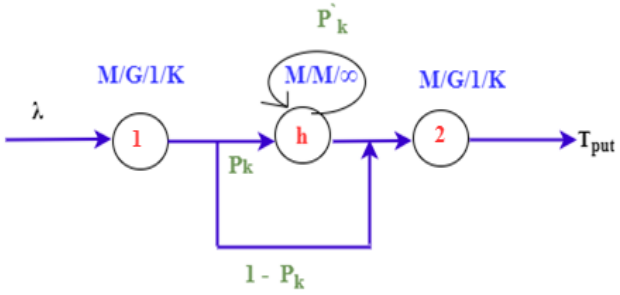

Fig.4. Expansion of the M/G/1/K as per Generalized Expansion Method

$$P_K = \frac{\rho^{\left(\frac{\sqrt{\rho e^{-s^2}}s^2-\sqrt{\rho e^{-s^2}}+2K}{2+\sqrt{\rho e^{-s^2}}s^2-\sqrt{\rho e^{-s^2}}}\right)}(\rho-1)}{\rho^{\left(2\frac{\sqrt{\rho e^{-s^2}}s^2-\sqrt{\rho e^{-s^2}}+K+1}{2+\sqrt{\rho e^{-s^2}}s^2-\sqrt{\rho e^{-s^2}}}\right)_{-1}}} \quad (20)$$

As shown in Figure 5, the artificial node is inserted between node1 and node2. The equations (9) through (14) needs to be applied to get the value of the desired variables. In case node2 is blocked, the arriving job is serviced at the holding node for the remaining service time interval of the customer in service. The delay distribution of a blocked customer at the holding node has the same distribution as node2.

$$\mu_h = \frac{2\mu_1}{1+s^2\mu_1^2} \quad (21)$$

where s^2 denotes the service time variance.

Expression (16) and expression (17) need to be applied to apply the feedback elimination. The arrival rate to node2 can be expressed as:

$$\lambda_j = \bar{\lambda_1}(1-P_K)/\alpha_j \quad (22)$$

$$\lambda_j = \bar{\lambda_1} - \lambda_h \quad (23)$$

Solving the equations (18), (9)–(14), (21), and (22) lead to the determination of all the variables of interest, including blocking probability and waiting time.

The response time of the systems can be determined using Little's law. Let P_k denote the steady-state probability of k ($0< k \leq K$) jobs in the queue.

$$P_K = \rho^K \times P_0 \quad (24)$$

where P_0 is the probability of zero jobs in the queue and K is the system buffer capacity. Number of jobs in the system L_s is equivalent to the summation of the product of the steady-state probabilities of the jobs. By using the steady-state probability distribution, the effective arrival rate of the jobs is:

$$\lambda_{eff} = \lambda^*(1-P_K) \quad (25)$$

The unconditional probability of empty buffer for a node having generalized service time distribution can be determined using (MacGregor Smith, 2011):

$$P_0 = \frac{(\rho-1)}{\rho^{\left(2\frac{\sqrt{\rho e^{-s^2}}s^2-\sqrt{\rho e^{-s^2}}+K+1}{2+\sqrt{\rho e^{-s^2}}s^2-\sqrt{\rho e^{-s^2}}}\right)_{-1}}} \quad (26)$$

Solving the equations (8) to (11) simultaneously; applying Little's formula (number of jobs in the queue is a product of arrival rate and queue waiting time) will lead to determination of all the variables of interest including queue waiting time, number of jobs in the system, effective arrival rate, and response time of the system.

Conclusion

The importance of IaaS cloud benchmarking has increased proportionately to the increased adoption of this technology by small and medium-sized businesses and scientific HPC users. In contrast to the fragmented state of the field today, we discuss a more focused, unified approach to IaaS benchmarking in this work, in which the community can collaborate on identifying the primary challenges and then sharing best practises and experiences.The available analytical approaches based on queuing theory are mathematically complex and time-consuming, especially when the generalised service distribution is assumed. GEM, which decomposes complex network scenarios into simpler and solvable open queuing networks, is used in this paper. In comparison to other analytical models, it has been demonstrated that the current methodology ensures a faster evaluation rate. Scientific workflows are becoming more prevalent on IaaS clouds, and resource provisioning has been a significant area of research for cost/performance optimization of the workflows. We review related work on this issue and the underlying research issues that need to be addressed in this paper.

REFERENCES

Adhikari, M., & Srirama, S. N. (2019). Multi-objective accelerated particle swarm optimization with a container-based scheduling for Internet-of-Things in cloud environment. *Journal of Network and Computer Applications*, *137*, 35–61.

Amato, F., Moscato, F., Moscato, V., & Colace, F. (2018). Improving security in cloud by formal modeling of IaaS resources. *Future Generation Computer Systems*, *87*, 754–764.

Ardagna, D., Casale, G., Ciavotta, M., Pérez, J. F., & Wang, W. (2014). Quality-of-service in cloud computing: modeling techniques and their applications. *Journal of Internet Services and Applications*, *5*(1), 1–17.

Asadi, A. N., Azgomi, M. A., & Entezari-Maleki, R. (2019). Evaluation of the impacts of failures and resource heterogeneity on the power consumption and performance of IaaS clouds. *The Journal of Supercomputing*, *75*(5), 2837–2861.

Ataie, E., Entezari-Maleki, R., Rashidi, L., Trivedi, K. S., Ardagna, D., & Movaghar, A. (2017). Hierarchical stochastic models for performance, availability, and power consumption analysis of IaaS clouds. *IEEE Transactions on Cloud Computing*, *7*(4), 1039–1056.

Brebner, P. C. (2012). Is your cloud elastic enough? Performance modelling the elasticity of infrastructure as a service (IaaS) cloud applications. *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, 263–266.

Carvalho, M., Menascé, D. A., & Brasileiro, F. (2017). Capacity planning for IaaS cloud providers offering multiple service classes. *Future Generation Computer Systems*, *77*, 97–111.

Cloud, F., Moreno-vozmediano, R., Montero, R. S., & Llorente, I. M. (2012). *Architecture : From Virtualized Datacenters to Infrastructures*. (December), 65–72.

Comput, J. P. D., Li, J., Qiu, M., Ming, Z., Quan, G., Qin, X., & Gu, Z. (2012). Online optimization for scheduling preemptable tasks on IaaS cloud systems. *J. Parallel Distrib. Comput.*, *72*(5), 666–677. https://doi.org/10.1016/j.jpdc.2012.02.002

Entezari-Maleki, R., Sousa, L., & Movaghar, A. (2017). Performance and power modeling and evaluation of virtualized servers in IaaS clouds. *Information Sciences*, *394*, 106–122.

Farid, M., Latip, R., Hussin, M., & Abdul Hamid, N. A. W. (2020). A survey on QoS requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing. *Symmetry*, *12*(4), 551.

Ghosh, R., Longo, F., Frattini, F., Russo, S., & Trivedi, K. S. (2014). Scalable analytics for IaaS cloud availability. *IEEE Transactions on Cloud Computing*, *2*(1), 57–70.

Ghosh, R., Longo, F., Naik, V. K., & Trivedi, K. S. (2013). Modeling and performance analysis of large scale IaaS clouds. *Future Generation Computer Systems*, *29*(5), 1216–1234.

Iosup, A., Prodan, R., & Epema, D. (2014a). *IaaS Cloud Benchmarking : Approaches , Challenges , and Experience*. 83–104. https://doi.org/10.1007/978-1-4939-1905-5

Iosup, A., Prodan, R., & Epema, D. (2014b). Iaas cloud benchmarking: approaches, challenges, and experience. In *Cloud Computing for Data-Intensive Applications* (pp. 83–104). Springer.

Jhawar, R., & Piuri, V. (2012). Fault tolerance management in IaaS clouds. *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*, 1–6. IEEE.

Labetoulle, J., & Pujolle, G. (1980). Isolation Method in a. *IEEE Transactions on Software Engineering*, *6*(4), 373–381.

Liu, B., Chang, X., Han, Z., Trivedi, K., & Rodríguez, R. J. (2018). Model-based sensitivity analysis of iaas cloud availability. *Future Generation Computer Systems*, *83*, 1–13.

MacGregor Smith, J. (2011). Properties and performance modelling of finite buffer M/G/1/K networks. *Computers and Operations Research*, *38*(4), 740–754. https://doi.org/10.1016/j.cor.2010.08.014

Manvi, S. S., & Krishna, G. (2013). Journal of Network and Computer Applications Resource management for Infrastructure as a Service (IaaS) in cloud computing : A survey. *Journal of Network and Computer Applications*, 1–17. https://doi.org/10.1016/j.jnca.2013.10.004

McDole, A., Abdelsalam, M., Gupta, M., & Mittal, S. (2020). Analyzing cnn based behavioural malware detection techniques on cloud iaas. *International Conference on Cloud Computing*, 64–79. Springer.

Strumberger, I., Bacanin, N., Tuba, M., & Tuba, E. (2019). Resource scheduling in cloud computing based on a hybridized whale optimization algorithm. *Applied Sciences*, *9*(22), 4893.

Zhou, A. C., & He, B. (2014). *Simplified Resource Provisioning for Workflows in IaaS Clouds*. https://doi.org/10.1109/CloudCom.2014.129