Journal Homepage: - www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI: 10.21474/IJAR01/21209
DOI URL: http://dx.doi.org/10.21474/IJAR01/21209

RESEARCH ARTICLE

A REVIEW AND COMPARATIVE STUDY ON TASK SCHEDULING IN GROUP MUTUAL EXCLUSION ALGORITHMS TO SOLVE CRITICAL SECTION PROBLEM BASED ON CLOUD COMPUTING

Pawan K. Thakur[1] and Vivek Chaudhary[2]

1. Associate Professor, Department of Computer Science and Engineering, Govt. College Dharamshala, H.P. (India).
2. Research Scholar, Career Point University, Kota, Rajasthan.

Manuscript Info

Manuscript History
Received: 16 April 2025
Final Accepted: 19 May 2025
Published: June 2025

Key words:-
Critical Section, Scheduling Methods, Cloud Computing, Quality of Service, Group Mutual Exclusion

Abstract

In large distributed systems which are based on cloud computing, the resources are shared to the clients. There must be some effective task scheduling method which efficiently use the different resources in cloud computing. In most of the algorithms which are based on group mutual exclusion , First come First serve scheduling method is used. But some others scheduling method are also used. These are round robin , priority scheduling and Johnson sequencing task scheduling and priority based job scheduling. All these are having some advantaged and disadvantages considering the different factors such as SLA , QoS and fault tolerance. In this paper , we present a review and comparative study of different scheduling algorithms which best suits for cloud computing.

"© 2025 by the Author(s). Published by IJAR under CC BY 4.0. Unrestricted use allowed with credit to the author."

Introduction:-

Cloud computing model provide on-demand network access to shared resources[1]. The services of cloud computing are hosted on a series of virtual machines running over the physical machines. The property of cloud elasticity must be fulfilled. The cloud elasticity is the ability to provide the cloud resources to different processes dynamically. The needs of customers for cloud services are shaped by various aspects, including deadlines, budgetary factors, payment rates, initiation times, duration of execution, and the required quantity of virtual machines. Effective cloud computing involves handling several applications simultaneously and efficiently distributing a range of resources. Resource management systems are essential for distributing resources among different applications, reclaiming resources from finished tasks, and enhancing their deployment to satisfy demand. [2]. Cloud service providers (CSPs) carefully implement resource management techniques, as resources like RAM, memory, processors, I/O devices, extra data centers. As a result, a pay-per-use model is used to provide users with designated amounts of resources, helping to avoid both underutilization and overutilization of resources. To enhance resource utilization and system performance, cloud computing requires the use of effective scheduling strategies. [3]. Cloud service providers aim to ensure effortless access to proficient cooperatives that can support their services and improve the overall cloud infrastructure. Scheduling is aimed to optimize the delivery of cloud services, taking into account the complexities of resource distribution, application deployment, and the fluctuating nature of user requirements. Cloud computing has transformed how customers engage with different cloud tiers, enabling them to implement

Corresponding Author:- Pawan K. Thakur
Address:- Associate Professor, Department of Computer Science and Engineering, Govt. College Dharamshala, H.P. (India).

applications. [4]. A crucial component of this system is the cloud broker, which serves as a platform to gather user data, analyze it, and communicate with Cloud Service Providers (CSPs) on behalf of clients while also providing billing solutions. The cloud broker's ability to integrate information can be effortlessly incorporated into any cloud networking, allowing users to oversee the execution times of their requests, monitor resource usage, and evaluate waiting times. Distributed computing has developed into a virtualized model where applications run transparently, despite the complexities of the cloud infrastructure. [5]. It provides flexible resource allocation and a strong platform to tackle multiple issues, such as effective request handling within a pay-as-you-go framework. [6]. Its dependability, ability to scale, and affordability, cloud computing has become extremely popular in addressing a variety of computational issues. [7]. In cloud computing, services are delivered to clients based on a shared understanding between the client and the Cloud Service Provider (CSP). These services are carried out through a series of tasks, leading to the idea of re-serving, where tasks may be redistributed for maximum efficiency. By tackling challenges such as delays in processing clinical requests and optimizing processor and resource use, the scheduling aims to provide valuable insights for advancing cloud computing practices, resulting in better service quality and increased customer satisfaction. Task scheduling in cloud computing is a complex computational challenge known as NP-Complete [8]. The goal of task scheduling is to optimize certain parameters like resource utilization, and power consumption by establishing the sequence in which tasks are performed on virtual machines. Companies that specialize in cloud services utilize various types of machines in their data centers to deliver timely services.

Objective of the Study:-

In this paper, we have presented a comparative study of various task scheduling algorithms in the context of cloud computing. The different algorithms used for task scheduling are First-Come-First-Serve (FCFS), Round Robin, and Priority Scheduling .

Related Work:

The problem of GME was firstly given by Yuh-Jeer Joung[9]. Joung proposed two different algorithm for GME. These are Joung's broadcast based algorithm[10] and Joung's quorum based algorithm[11]. Joung's broadcast based algorithm was an extension of Ricart and Agarwala distributed mutual exclusion algorithm[12]. Joung proposed two algorithms RA1 and RA2. In RA1, the process which wants to enter the critical section , sends a request message to all the processes and upon receiving reply message from all the processes, it enters the critical section. There are some concurrency related issues in RA1, which was later solved by using RA2. In Joung's quorum based algorithm , the concept of quorum is used. A process has to obtain permission from all the processes in the quorum to enter critical section. For concurrency, Joung proposed two algorithms , the first one is Maekawa_M, which sends message in parallel and second one is serial version called Maekawa_S, which obtains sequential permission from each process in quorum. These two algorithms avoids deadlock .

In comparison to classical distributed systems , the working in cloud computing is different because it deals with different characteristics . The different characteristics in cloud computing includes fault-tolerance, QoS, scalability and priority. There are different priority based algorithms which are used for real time systems. These can be categorized as :

(i)Static priority algorithms (ii) Dynamic priority algorithms.

The priority in static priority algorithms remains the same. There is no priority inversions but it can lead to starvation as low priority processes cannot be able to enter the critical section. Housni and trehel[13] proposed an algorithm where sites with same priority forms the group. It uses router for external communication and the processes within the group communicate with each other by passing messages. When any process wants to enter the critical section, it sends the request and that request is forwarded to the root. The root sends the token request to the routers. In each group , the Raymond algorithm[14] is used.

In dynamic priority algorithms, the priority of algorithm is increased with the passage of time. For increasing the priorities , different factors such as request time, level and distance are used in different algorithms. In Kanrar-Chaki[15] token based algorithm , which is based on Raymond algorithm[14], the low priorities of pending requests are increased dynamically. In avoids starvation but increases priority inversions. Jonathan Lejeunl et al[16] proposed a token based algorithm where new concepts have been added in Kanrar-Chaki[15] token based algorithm. These are level heuristics and level distance heuristics. Level heuristics postpones the priority increment of pending requests. In level distance heuristics , the processes are incremented according to the level of the tree. These two heuristics removes the drawbacks of the Kanrar-Chaki[15] token based algorithm where the low priority processes

frequently access the critical section which is priority inversion. In priority inversions, a low priority process has been granted the access to critical section before the high priority process which is violation of Service Level Agreement. Jonathan Lejeunl et al[16] proposed a new algorithm where the attempt is balance the priority inversions and response time of low priority processes. It uses the awareness concept which aims at reducing maximum response time whereas the number of priority inversions remains low. For this global view of pending requests is necessary.

Task scheduling plays a vital role in distributed computing settings, especially within cloud computing. Efficient scheduling techniques strive to reduce task waiting periods and improve overall cloud performance to maximize advantages. The goal of utilizing different scheduling algorithms is to determine a suitable task sequence that shortens the total execution time. Considering the distributed and diverse characteristics of cloud environments, conventional scheduling algorithms may not be suitable for direct application. Therefore, it is important to create scheduling algorithms designed specifically for cloud systems. [17]. By tackling the specific challenges presented by cloud environments, these tailored scheduling algorithms can improve resource management, decrease latencies, and enhance overall system performance, ultimately resulting in greater advantages for both cloud service providers and users. Effective task scheduling is vital for unlocking the complete potential of cloud computing and fulfilling the increasing demands of various applications and services in the digital age. As researchers, it is essential to investigate new and efficient scheduling algorithms designed for cloud environments to continually improve cloud services and promote progress in the field of distributed computing. In the field of virtual machine (VM) selection for application scheduling, Naik et al. [18] introduced a novel hybrid multiobjective heuristic method that combines the Non-dominated Sorting Genetic Algorithm-2 (NSGA-II) with the Gravitational Search Algorithm (GSA). This hybrid strategy aims to improve both the efficiency and efficacy of the scheduling process by leveraging the advantages of NSGA-II and GSA. While GSA focuses on utilizing effective solutions to search for optimal results and avoid becoming stagnant, NSGA-II expands the exploration range through a thorough investigation. The main goal of this hybrid approach is to achieve better job scheduling performance, concentrating on three essential factors: maximizing the total number of scheduled jobs, reducing overall energy consumption, and achieving the shortest response time alongside the lowest cost. It is crucial to recognize that current scheduling algorithms in VMs do not cater to the specific needs and goals that this hybrid approach considers. Consequently, the suggested integration of NSGA-II and GSA presents an innovative and promising method for tackling the challenges associated with VM selection and application scheduling, which may enhance cloud computing performance and resource efficiency. In the field of public cloud computing, a variety of heuristic algorithms have been created and utilized to efficiently schedule a range of jobs. Among the most significant developments in heuristic methods are the First Come, First Serve (FCFS) algorithm, the Min-Max algorithm, the MinMin algorithm, and the Suffrage computation. Furthermore, other notable innovations in this area include Greedy Scheduling, Shortest Task First (STF), Sequence Scheduling, Balance Scheduling (BS), Opportunistic Load Balancing, and Min-Min Opportunistic Load Balancing [19], [20], [21]. These heuristic algorithms are vital for task scheduling within the public cloud, focusing on optimizing various performance metrics such as job completion times, resource utilization, and system effectiveness. Each algorithm tackles the scheduling challenge from a unique angle, applying specific rules and strategies to meet the intended goals.

Analysis of different Scheduling algorithms:

First Come First Serve algorithm:

The First-Come-First-Served (FCFS) scheduling algorithm functions by executing tasks in the sequence they are received, employing a non-preemptive strategy. The average waiting time and overall turnaround time for tasks are affected by their size and the timing of their arrival. In a cloud computing setting, several clients seek resources from the data center controller, and these requests are sent to the FCFS virtual machine load balancer. The FCFS virtual machine load balancer processes tasks according to the order in which client requests arrive, [22], [23], [24]. The FCFS approach has been extensively researched and utilized in cloud computing because of its simplicity and equitable resource distribution based on arrival times. Nonetheless, it may result in suboptimal resource usage and increased wait times for tasks of different sizes and priorities. To mitigate these drawbacks, scholars have investigated alternative task scheduling methods, such as Round Robin, Priority Scheduling, and Johnson Sequencing, each presenting unique benefits and specific strategies to enhance cloud resource management and performance.

Priority Scheduling algorithm:

The Priority Scheduling algorithm functions by executing tasks according to their designated priorities, with tasks of higher priority being processed before those of lower priority. This scheduling method is often utilized in operating systems that manage numerous tasks, where the order of execution is influenced by their priority levels. Priority Scheduling can also be applied as a preemptive strategy, enabling a higher priority task to interrupt and take over the execution of tasks with lower priority, [25], [26], [27], [28]. The priority-driven method of task scheduling is beneficial for real-time systems and applications that require certain tasks to be prioritized based on their importance or urgency. However, implementing priority scheduling may result in challenges such as starvation, where lower priority tasks experience significant delays in being executed. It is crucial to strike a balance in priority levels and take task attributes into account to ensure equitable resource distribution and avoid scenarios where low-priority tasks are indefinitely postponed. As the study of task scheduling in cloud computing progresses, it is important to investigate the performance of different scheduling algorithms, including Priority Scheduling, across various circumstances, workload distributions, and system setups.

Priority Scheduling Algorithm, follows these steps:

• Initialization: The process starts by setting up the list of tasks along with their respective priorities. Each task is depicted as a job or process, with its priority determined by established criteria, including the significance of the task, deadline requirements, or preferences set by the user.

• Sort Tasks: The subsequent step involves arranging the list of tasks according to their priorities, with higher priorities listed first. This arrangement guarantees that tasks with greater importance are positioned at the beginning of the list, while those of lesser importance are located towards the end.

• Execution: The algorithm executes tasks based on their priority levels. The task that has the highest priority is chosen first for execution. The manner of execution can differ based on whether the algorithm operates in a preemptive or non-preemptive manner. In a preemptive algorithm, a currently executing task can be interrupted if a task with a higher priority arrives. The system continuously monitors for any higher priority tasks that may arise during the execution of a task. If a higher priority task is detected, the current task is interrupted, and the higher priority task is scheduled to run. In non-preemptive mode, the current task is permitted to finish executing before selecting and scheduling the next task with the highest priority.

• Task Completion: After a task is finished, the algorithm moves on to the subsequent task following the established priority sequence. This cycle continues until all tasks have been carried out.

• Task Arrival: When carrying out tasks, it is possible for new tasks to enter the system. In the case of a preemptive algorithm, the priority of the new task is assessed against the priority of the task currently in progress. If the new task's priority is greater, it interrupts the ongoing task, and the new task is then scheduled for execution.

• Task Termination: When tasks finish executing, they are taken off the list, and the algorithm proceeds to choose the next task with the highest priority for execution.

• Completion Check: The algorithm carries on performing tasks until every task in the list has been finished. After all tasks have been executed, the scheduling procedure comes to an end.

Round Robin Scheduling algorithm:

The Round-Robin (RR) scheduling algorithm is a basic preemptive scheduling method used in different computing environments. In RR, each process is assigned a specific time quantum or time slice by the CPU. The processes are managed in a First-Come-First-Serve (FCFS) order, allowing them to run for the length of the time quantum. After the time quantum expires, the current process is interrupted, and the CPU shifts to the next process in line. This preemption and switching between tasks persist until all processes in the system have finished their execution. The RR scheduling algorithm is commonly utilized in operating systems and distributed computing settings due to its straightforwardness and equitable resource distribution. It guarantees that every task receives an equitable portion of the CPU's time, thereby preventing any single task from dominating the CPU for too long. By implementing a fixed time quantum, RR achieves a balance between responsiveness and efficiency in executing tasks. The main characteristics of the Round-Robin scheduling algorithm include the following:

• Preemptive Scheduling: RR functions as a preemptive scheduling algorithm, enabling tasks to be interrupted and rescheduled even if they haven't completely used their time quantum. This capability facilitates a flexible and responsive distribution of resources.

Time Quantum: The time quantum is an essential factor in the RR algorithm. It defines the duration for which each task can execute before being interrupted. Selecting the right time quantum affects the trade-off between system responsiveness and the overhead caused by context switching.

• FCFS Order: Tasks are organized in a queue according to their arrival time, and the RR algorithm executes them following the FCFS sequence. This guarantees that tasks are processed in the order of their arrival, promoting fairness in the distribution of resources.
• Preemption Handling: When a task's time slice runs out, the processor records its current state and transitions to the following task in the queue. The interrupted task is then placed at the end of the line to wait for its next opportunity.

The Round-Robin scheduling algorithm offers a viable method for organizing tasks across different computing settings. Nevertheless, its efficiency can be affected by factors such as the selected time quantum, the characteristics of the tasks being processed, and the total system load. Ongoing research is dedicated to investigating modifications and improvements to RR to boost its performance and flexibility in various situations [29], [30], [31].

Dynamic Heuristic Johnson Sequencing Algorithm :
The Dynamic Heuristic Johnson Sequencing (DHJS) algorithm presents an innovative method that integrates the calculation of dynamic burst time and the Johnson sequencing technique to enhance task scheduling in a multi-server setting. Initially, the DHJS algorithm computes the dynamic time quantum for the tasks using the median burst time and the maximum burst time of the tasks. This time quantum is then implemented in a Round Robin scheduling method. Following this, the Johnson sequencing algorithm is utilized to establish the optimal order for executing the tasks.

The DHJS algorithm offers a flexible and heuristic method to tackle intricate task scheduling problems in multi-server settings. By integrating burst time calculation, Johnson sequencing, and queuing model evaluation, it seeks to enhance task execution and resource distribution, resulting in greater efficiency and prompt task completion for customers. Continued research and testing are recommended to confirm and improve the effectiveness of the DHJS algorithm across different practical cloud computing situations.

Comparative study of different algorithms:
Based on factors:

Sr.No	Scheduling Algorithm	Factors	Advantages	Disadvantages
1	First Come First Serve algorithm	Minimum waiting time	Increases average response time	Does not consider additional factors
2	Priority Algorithm	Task priority	For task scheduling , priority is considered.	Sometime lack of consistency and complex structure.
3	Round Robin Algorithm	Arrival time	Fairly distributed loads and easy to implement	Preemption is required.
4	DYNAMIC HEURISTIC JOHNSON SEQUENCING ALGORITHM :	Arrival time, Turnaround time	Minimizes the service time in cloud computing and high performance.	Complexity

Based on scheduling methods:

Scheduling method	Job scheduling	Static scheduling	Dynamic scheduling	Cloud Environment
First Come First Serve algorithm	Yes	Yes		Yes
Priority Algorithm	Yes	No	Yes	Yes
Round Robin Algorithm	Yes	No	Yes	Yes
DYNAMIC HEURISTIC JOHNSON SEQUENCING ALGORITHM :	Yes	No	Yes	Yes

Conclusion:-

The first come first serve algorithm is fair for simple applications involving less complex structure. The priority algorithm takes into account the priority based on dynamic scheduling. The round robin suits for applications where time quantum is used, All these three algorithms do not fulfils the adaptive nature of the cloud computing. The Dynamic heuristic Johnson sequencing algorithm best suits for scheduling the jobs in cloud computing. It minimizes service time and increases the performance. Considering the dynamic nature , it schedule the jobs in fairly manner,

References:-

[1].Edmondson, J., Schmidt, D., & Gokhale, A. (2011). QoSenabled distributed mutual exclusion in public clouds. On the Move to Meaningful Internet Systems: OTM 2011, 542-559.
[2].X. Li, T. Jiang, and R. Ruiz, ‘‘Heuristics for periodical batch job scheduling in a MapReduce computing framework,’’ Inf. Sci., vol. 326, pp. 119–133, Jan. 2016.
[3].Y. Xiong, S. Huang, M. Wu, J. She, and K. Jiang, ‘‘A Johnson’s-rule-based genetic algorithm for two-stage-task scheduling problem in data-centers of cloud computing,’’ IEEE Trans. Cloud Comput., vol. 7, no. 3, pp. 597–610, Jul. 2019.
[4].Z. Pan, X. Hou, H. Xu, L. Bao, M. Zhang, and C. Jian, ‘‘A hybrid manufacturing scheduling optimization strategy in collaborative edge computing,’’ Evol. Intell., vol. 3, pp. 1–13, Oct. 2022.
[5].A. Lachmann, D. Torre, A. B. Keenan, K. M. Jagodnik, H. J. Lee, L. Wang, M. C. Silverstein, and A. Ma’ayan, ‘‘Massive mining of publicly available RNA-seq data from human and mouse,’’ Nature Commun., vol. 9, no. 1, p. 1366, 2018.
[6].S. O. Bukhari, ‘‘Cloud algorithms: A computational paradigm for managing big data analytics on clouds,’’ in Intelligent Systems. Singapore: Springer, 2021, pp. 455–470.
[7].Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, ‘‘A novel task scheduling scheme in a cloud computing environment using hybrid biogeographybased optimization,’’ Soft Comput., vol. 23, no. 21, pp. 11035–11054, Nov. 2019.
[8]. A. Wilczyński and J. Kołodziej, ‘‘Modelling and simulation of securityaware task scheduling in cloud computing based on blockchain technology,’’ Simul. Model. Pract. Theory, vol. 99, Feb. 2020, Art. no. 102038.
[9]. Y.-J. Joung, ,”The congenial talking philosophers problem in computer networks”, Distributed computing Vol.15,pp 155-175,2002.
[10]. Y.-J. Joung, “ Quorum based algorithm for group mutual exclusion”, IEEE transactions on parallel and distributed system, vol 14, no. 5 pp.2003,may2003
[11]. Y.-J. Joung, Asynchronous group mutual exclusion, Distributed Comput. (DC) 13 (4) (2000) 189–206.
[12] .Ricart, G., Agrawala, A.: An Optimal Algorithm for Mutual Exclusion in Computer Networks. CACM, Vol. 24(1). (1981) 9–17
[13]Housni, A., & Trehel, M. (2001). Distributed mutual exclusion token-permission based by prioritized groups. In Computer Systems and Applications, ACS/IEEE International Conference on. 2001 (pp. 253-259). IEEE.
[14]Raymond, K. (1989). A Tree-Based Algorithm for Distributed Mutual Exclusion. ACM Trans. Comput. Syst., 7, 61-77.
[15]KANRAR, S., & CHAKI, N. (2010). FAPP: A New Fairness Algorithm for Priority Process Mutual Exclusion in Distributed Systems. JNW, 5(1), 11-18.
[16]Lejeune, J., Arantes, L., Sopena, J., & Sens, P. (2012, May). Service level agreement for distributed mutual exclusion in cloud computing. In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012) (pp. 180-187). IEEE Computer Society.
[17]E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, ‘‘Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends,’’ Swarm Evol. Comput., vol. 62, Apr. 2021, Art. no. 100841.
[18]K. Naik, G. Gandhi, and S. Patil, ‘‘Multiobjective virtual machine selection for task scheduling in cloud computing,’’ in Computational Intelligence: Theories, Applications and Future Directions (Advances in Intelligent Systems and Computing). Singapore: Springer, 2019, pp. 319–331.
[19]M. A. Elaziz, S. Xiong, K. P. N. Jayasena, and L. Li, ‘‘Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution,’’ Knowl.-Based Syst., vol. 169, pp. 39–52, Apr. 2019
[20]D. Alsadie, ‘‘A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers,’’ IEEE Access, vol. 9, pp. 74218–74233, 2021
[21] Q. Zhang, L. Cheng, and R. Boutaba, ‘‘Cloud computing: State-of-the-art and research challenges,’’ J. Internet Services Appl., vol. 1, no. 1, pp. 7–18, May 2010.

[22]S. V. Angiuoli, M. Matalka, A. Gussman, K. Galens, M. Vangala, D. R. Riley, C. Arze, J. R. White, O. White, and W. F. Fricke, ‘‘CloVR: A virtual machine for automated and portable sequence analysis from the desktop using cloud computing,’’ BMC Bioinf., vol. 12, no. 1, pp. 1–15, Dec. 2011.
[23]S. More, S. Muthukrishnan, and E. Shriver, ‘‘Efficiently sequencing taperesident jobs,’’ in Proc. 18th ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst., Philadelphia, PA, USA, May 1999, pp. 33–43.
[24]S. Zhang, H. Yan, and X. Chen, ‘‘Research on key technologies of cloud computing,’’ Phys. Proc., vol. 33, pp. 1791–1797, Jan. 2012.
[25]M. Choudhary and S. K. Peddoju, ‘‘A dynamic optimization algorithm for task scheduling in cloud environment,’’ Int. J. Eng. Res. Appl., vol. 2, no. 3, pp. 2564–2568, 2012.
[26]R. K. R. Indukuri, S. V. Penmasta, M. V. R. Sundari, and G. J. Moses, ‘‘Performance evaluation of deadline aware multi-stage scheduling in cloud computing,’’ in Proc. IEEE 6th Int. Conf. Adv. Comput. (IACC), Feb. 2016, pp. 229–234.
27.S. Pal and P. K. Pattnaik, ‘‘A simulation-based approach to optimize the execution time and minimization of average waiting time using queuing model in cloud computing environment,’’ Int. J. Electr. Comput. Eng., vol. 6, no. 2, p. 743, Apr. 2016.
28.C. Sriskandarajah and S. P. Sethi, ‘‘Scheduling algorithms for flexible flowshops: Worst and average case performance,’’ Eur. J. Oper. Res., vol. 43, no. 2, pp. 143–160, Nov. 1989.
29.A. S. Prasad and S. Rao, ‘‘A mechanism design approach to resource procurement in cloud computing,’’ IEEE Trans. Comput., vol. 63, no. 1, pp. 17–30, Jan. 2014.
30. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, ‘‘CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,’’ Softw., Pract. Exp., vol. 41, no. 1, pp. 23–50, Jan. 2011.
[31] A. V. Karthick, E. Ramaraj, and R. G. Subramanian, ‘‘An efficient multi queue job scheduling for cloud computing,’’ in Proc. World Congr. Comput. Commun. Technol., Feb. 2014, pp. 164–166.