International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

Load Optimized M/M/M Queueing for Parallel Job Scheduling in Multiple Cloud Centers

C. Antony
Research and Development Centre:
Bharathiar University
Coimbatore, Tamil Nadu 641 046, India

C. Chandrasekar
Professor
Department of Computer Science
Periyar University
Salem, Tamil Nadu 636 011, India

***Abstract*— The cloud computing environment enables cloud users to execute their applications in remote data centers. Many of these applications are considered to be highly complex in nature, requiring parallel processing capabilities. Parallel job scheduling techniques, mainly focus on improving throughput or the information processed by the cloud center in a given interval of time and reducing average task waiting time. For a data center that deals with parallel jobs, it is required to design an optimal scheduler resulting in minimal utilization of memory rate for scheduling each job. In this paper, we present M/M/M Queuing System and Load Optimized (QS-LO) model for parallel job scheduling in multiple cloud centers. The QS-LO model is designed as two algorithms, namely parallel job scheduling and load balancing with aiming at improving the throughput rate and reducing the average task waiting time in multiple cloud centers. We perform experimental analysis using benchmarks and synthetic datasets to measure the performance of the proposed algorithm. The experimental results are compared with the existing parallel job schedulers in terms of the job assigned, throughput memory utilization rate and scheduler time interval. The experimental results show that the QS-LO model is able to improve the throughput rate and also reduce the average task waiting time when compared to the state-of-the-art works.**

Keywords—Cloud Computing; Parallel Job Scheduling; M/M/M Queuing System; Virtual Machine; Load balancing;

I. INTRODUCTION

With the advancement of computer and internet technologies, cloud computing environment has been successfully used in several information systems. Though, cloud computing environment provides a better way to accomplish the submitted jobs in terms of responsiveness and scalability, but most of the job scheduling problems in parallel cloud computing environments are still hard to design.

Hyper Heuristic Scheduling Algorithms (HHSA) [1] find scheduling solutions for cloud computing systems. The HHSA algorithm used two detection operators to determine when to change the low-level heuristic algorithm and a perturbation operator to fine-tune solutions with the objective of improving the scheduling results in terms of job request made. However, the computation cost and deadline constraints were not taken into consideration. To address these two issues, Global Greedy Budget and Gradual Refinement [2] were designed so that the solutions were gradually refined by combining dynamic programming techniques with a local greedy algorithm. Next, two algorithms GGB and GR were designed on the basis of greedy strategy by aiming at minimizing the time complexity in reducing the scheduling lengths of the workflows without breaking the budget. Though, scheduling, length and scheduling time was reduced, but the information processed by cloud center remains unaddressed.

In this paper, we investigate the problem of parallel job scheduling as a workflow within memory and throughput parameters. This workflow with the M/M/M queuing system handles multiple user requests in multiple cloud centers. Besides, we address load-optimized function, which is finely grained compared to the existing state-of-the-art methods.

The remainder of the paper is organized as follows. Section 2 begins with a brief introduction to the traditional job scheduling models to be designed in cloud computing environments and scheduling problem and its relevant technologies, Section 3 provides a brief discussion on M/M/M Queuing System and Load Optimized model. The simulation results on CloudSim are discussed in Section 4. Conclusions are drawn in Section 5.

II. RELATED WORKS

With the recent advancements in hardware for handheld mobile devices, resource-intensive applications still requires large computation and storage capabilities. Recent research works have addressed these issues by employing remote servers, such as clouds. An approach called k-out-of-n-computing [3] was employed to demonstrate the fault tolerance and energy efficiency in large scale networks. However, heterogeneous resource allocation remained unaddressed. In [4], a polynomial time to achieve the optimal throughput under resource constraints were addressed.

Cloud computing environment has received considerable attention due to its promising future and delivering various computing and IT services. Optimal power allocation and load distribution for multiple servers across cloud and data centers was designed in [5]. In [6], a pricing model called M/M/m queue model was designed with the objective of improving profit maximization in cloud computing. To perform efficient thermal aware scheduling in geographically distributed centers, online scheduling and optimal offline algorithm was designed in [7].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

In Cloud computing, a ubiquitous model offers distributed and elastic resources in the form of Virtual Machines (VMs). Uncertainty Based Task Scheduling (UBTS) was designed in [8] with the objective of reducing uncertainty and improving average cloud utilization. The load balancing algorithm based on ant colony [9] was designed to improve the workflow scheduling and to improve load balancing ability. However, optimization remains unaddressed. In [10], the dynamic resource allocation algorithm was designed to improve the performance in the situation where resource contention was observed to be fierce.

One of the business model for cloud computing is the distributed computing environment. An adaptive heuristic scheduling process was designed in [11] to avoid frequent allocation of specific server and to minimize the total execution time. The fair scheduling model was designed in [12] with an objective of minimizing the switching time. To achieve fair throughput, an analysis was made in [13] with the objective of providing high throughput processing. Another method based on map reduce was investigated in [14] by using multi algorithm execution to reduce the I/O cost and complexity. Nash equilibrium resource allocation based on game theory was designed in [15] to decrease the response time and complexity involved in it.

Efficient resource management is one of the key and remains the main operational goal in large-scale computer systems that includes cloud computing environment. In [16], Maximum Effective Reduction (MER) algorithm was designed to improve the resource efficiency while schedules in a cloud environment. Another method called bi-objective workflow scheduling was designed in [17] to improve the resource allocation on requests. A heuristic task scheduling algorithm [18] was designed using randomly generated graphs and set of task graphs to achieve best scheduling performance. Task scheduling optimization [19] was investigated to achieve better performance in terms of processing time while considering network contention and could cost. Another method to schedule parallel jobs using migration and consolidation was designed in [20] to reduce the average response time.

Adaptive-Scheduling-with-QoS-Satisfaction algorithm called as AsQ was presented in [22] for the hybrid cloud environment to improve the resource utilization rate of the private cloud and to reduce the task response time. The Ant Colony Optimization algorithm was designed in [23] to improve the task scheduling process by means of dynamically scheduling the tasks and to improve the throughput and quality of service (QoS) of Mobile Cloud Computing. A hybrid job scheduling approach was developed in [24] using genetic algorithm and fuzzy theory for considering the load balancing of the system and to diminish the total execution time and the execution cost.

III. PARALLEL JOB SCHEDULING

A queuing system using the M/M/M load optimized function for parallel job scheduling and efficient load balancing is introduced by evaluating servers, Poisson input, exponential service and load balancer to the cloud infrastructure. We begin by describing the problem specification for parallel job scheduling and then present the QS-LO model.

A. Problem Specification

In a cloud computing environment, parallel job scheduling is dependent on the effectiveness of the methods used to execute the job. In our work, the cloud computing environment is said to be assumed, to be hosted in a data center consisting of many cloud servers that provide a resource by virtual machines. The cloud servers and virtual machines may possess different memory sizes, processing capacities and response time.

Let us consider the data center $'DC = DC_1, DC_2, \ldots, DC_n'$ of cloud servers $'CS = CS_1, CS_2, \ldots, CS_n'$. And the assume cloud servers consist of many virtual machines in a cloud environment like $'CS_i = \{vm_{i1}, vm_{i2}, \ldots, vm_{in}\}$. The goal of QS-LO model is to schedule parallel jobs in the multi cloud center and to improve the information processed by cloud data by means of reducing the average task waiting time. In the next section we start with extending these results to the M/M/M Queuing System and Load Optimized function for parallel job scheduling.

B. M/M/M Queuing System and Load Optimized Function

To formulate and study the problem of parallel job scheduling and load balancing in multi cloud centers, we need a model for a queuing system and a design of virtual machines. Let us consider a parallel job scheduling that comprises of jobs represented by a Directed Acyclic Graph (DAG). The vertices of DAG denote the partitioned jobs of the corresponding application, whereas the edges of DAG denote preference between the jobs. Hence, the graph is represented as $'G = (N, E)$, that comprises of set $'N'$ of $'n'$ users and a set $'E'$ of $'e'$ edges respectively.

Let $'V = J_1, J_2, \ldots, J_n'$ represents the number of jobs to be executed in the datacenter $'DC'$. On the other hand, $'s = s_1, s_2, \ldots, s_n'$ represent the size of the job to be executed. The parallel job scheduling and load balancing model for $'n'$ virtual machines across multiple clouds and data centers is shown in Fig 1.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

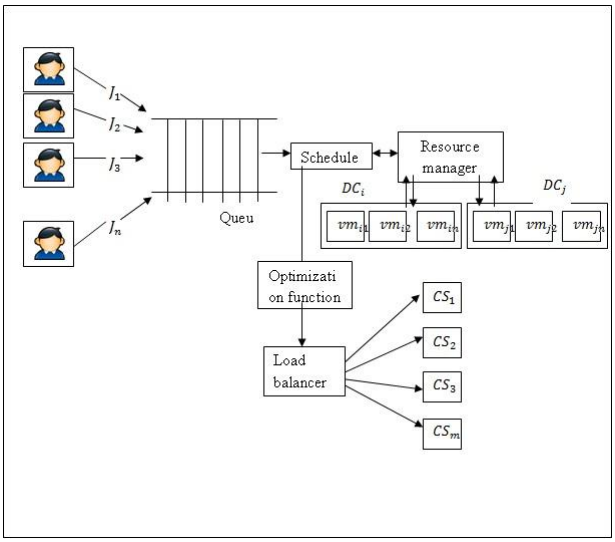

Fig. 1. Structure of Queuing System and Load Optimized parallel job scheduling

Fig. 1 shows the structure of the M/M/M Queuing System and Load Optimized parallel job scheduling in multiple cloud centers. As shown in the figure, Landsat dataset on AWS is given as input to the QS-LO model. To start with, cloud users in cloud environment place their requests to be processed. These jobs are placed in a queue. The jobs arrived comprises of either independent or dependent jobs. In case of independent jobs, arrived job is given as input to the scheduler. On the other hand, if the job is said to be dependent, dependent jobs are notified to the scheduler so that the parent jobs are scheduled after child jobs are executed.

The dependent jobs contain the jobs that depend on the other jobs present in the virtual machines. Finally, jobs are assigned as the input to the scheduler. In certain cases, the jobs may take the execution time more than the allotted, resulting in overload. In such situation, resource manager identifies the unused virtual machines. If there appears to be no unused virtual machine, then no job migration takes places. On the contrary, job migration takes place when the resource manager finds any unused virtual machine. In this way, an efficient scheduler and load balancer is designed.

C. Construction of M/M/M Queuing System

In this section, an M/M/M queuing system with multiple job arrivals to optimize the performance in a cloud computing environment is investigated. In a cloud computing environment, multiple requests are placed by cloud users and the cloud server acts as a single point of access for all types of cloud users. The cloud server is a collection of cloud server resources that is provided by the cloud provider to host all the applications for the cloud users.

In order to construct M/M/M queuing system, let us assume that there are 'n' job requests and 'm' cloud servers in a cloud computing environment. As the job arriving requests may be sent from different cloud users, the inter-arrival time is a Poisson process with job arrival rate '$J_i\lambda_i$'. Job requests from different cloud users in the scheduler's queue are distributed to different cloud servers with the rate of scheduling depending on the scheduler. Let us consider that there are 'm' cloud servers '$CS_1, CS_2, \ldots, CS_m$' in datacenter, then the total job arrival rate is as given below.

$$\lambda = \sum_{i=1}^{n} J_i \lambda_i \quad (1)$$

On the other hand, the total cloud server's service rate is as given below.

$$\mu = \sum_{i=1}^{m} J_i \mu_i \quad (2)$$

As the cloud users' requests come from all over the world, the cloud computing environment provides infinite services, without limiting the source of cloud users and the number of the queuing model. If If '$\rho = \lambda/\mu$', then the steady state equations for the M/M/M Queuing System are given as below.

$$\left.\begin{aligned} \rho_1 &= m\rho P_0 \\ \rho_2 &= \frac{m^2}{2!}\rho^2 P_0 \\ \rho_3 &= \frac{m^3}{3!}\rho^3 P_0 \\ \rho_m &= \frac{m^m}{m!}\rho^m P_0 \end{aligned}\right\} \quad (3)$$

Where

$$P_0 = \left[1 + \left(\frac{\lambda}{\mu}\right) + \left(\frac{\lambda}{\mu}\right)^2 + \left(\frac{\lambda}{\mu}\right)^3 + \ldots + \left(\frac{\lambda}{\mu}\right)^k\right]$$

Hence, $$P_0 = \frac{1}{\sum_{n=0}^{k}\left(\frac{\lambda}{\mu}\right)^n} \quad (4)$$

Once, the total job arrival rate 'λ' and total cloud server's service rate 'μ' are obtained, independent job and dependent job forms as input to the scheduler. The job scheduler selects the suitable virtual machine based on QLS algorithm. The job scheduler gathers the resource information from the resource manager. It applies the proposed algorithm to find the suitable virtual machine for the given job. The load for single machine 'L_{sm}' is as given below.

$$L_{sm} = \frac{L}{\sum_{i=1}^{n} c_i} \quad (5)$$

From (5), 'L' represents the load of the overall node (i.e. virtual machine) in cloud computing environment, on the other hand, 'c_i' represents the capacity of a single node. Fig. 2 shows the Queue-based Load Scheduling algorithm.

Input: Data Center '$DC = DC_1, DC_2, \ldots, DC_n$', Cloud Server '$CS = CS_1, CS_2, \ldots, CS_m$', Virtual Machine '$\{VM = vm_{i1}, vm_{i2}, \ldots, vm_{in}\}$', Jobs '$J = J_1, J_2, \ldots, J_n$', Job size '$s = s_1, s_2, \ldots, s_n$', Threshold '$T$'
Output: Improved throughput rate
1: Begin 2: For each job request 'J'

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

3: Measure the job arrival rate using (1)
4: Measure the cloud server's service rate using (2)
5: Obtain steady state using (3)
6: Measure load for single machine using (5)
7: If '$VM < T - \sum_{vm=1}^{k} L_{vm}$', then
8: Virtual machine is under-loaded
9: Assign job to the VM
10: End if
11: If '$VM > T - \sum_{vm=1}^{k} L_{vm}$', then
12: Virtual machine is over-loaded
13: Assign job to other VM
14: End if
15: If '$VM = T - \sum_{vm=1}^{k} L_{vm}$', then
16: Virtual machine is balanced
17: Assign job to the VM
18: End if
19: End for
20: End

Fig. 2. Queue-based Load Scheduling algorithm

As shown in the Fig. 2, for each job request from cloud user, the job arrival rate and cloud server's service rate are measured. Followed by this, steady state equation is obtained based on the total job arrival rate 'λ' and total cloud server's service rate. Finally, load is measured by assigning a threshold factor 'T' to allocate the job to the appropriate virtual machine 'VM' which in turn improves the throughput rate of QS-LO model in an effective manner.

D. Load-Based Optimization Function

Once the queue-based load scheduling is performed in a parallel manner in a cloud environment, the next step is to optimize the performance of multiple job requests in a cloud computing environment. The proposed work uses job utilization rate, average time and the number of cloud users waiting for servicing at each cloud server as the optimization metrics.

The Load-based Optimization Function is given as below. All the cloud users' job request arrives at the scheduler, which selects the job request to the cloud server according to the result of the load-based optimization function. The input matrix of the load-based optimization function is the average time, job utilization rate, and the total number of cloud users (i.e. Job) waiting for each cloud server in the cloud computing environment.

Let us assume that '$Prob_i$' denotes the probability of 'i' cloud user to be selected and '$Time_i$' represents the cloud users servicing time. Then, the average time '$AvgTime$' is as given below.

$$AvgTime = \sum_{i=1}^{n} Prob_i Time_i \quad (6)$$

The job utilization rate 'U_i' is measured as given below.

$$U_i = J_i \frac{\lambda_i}{\mu_i} \quad (7)$$

Where 'λ_i' represents the total job arrival rate and 'μ_i' represents the total cloud server's service rate. Finally, the optimization function '$OpFn$' is measured as given below.

$$OpFn = AvgTime * U_i * J_i \quad (8)$$

With the resultant average time (6) and job utilization rate (7), optimization function (8) is evolved. Fig. 3 shows the load-based optimization algorithm.

Input: Data Center '$DC = DC_1, DC_2, \ldots, DC_n$', Cloud Server '$CS = CS_1, CS_2, \ldots, CS_m$', Virtual Machine '$\{VM = vm_{i1}, vm_{i2}, \ldots, vm_{in}\}$', Jobs '$J = J_1, J_2, \ldots, J_n$' probability of '$i$' cloud user to be selected '$Prob_i$', cloud users servicing time '$Time_i$' total job arrival rate 'λ_i', total cloud server's service rate 'μ_i'

Output: Minimized average job waiting time

1: Begin
2: For each Jobs 'J' and Cloud Server 'CS'
3: Measure average time using (6)
4: Measure job utilization rate using (7)
5: Measure optimization function using (8)
6:End for
7: End

Fig. 3. Load-based optimization algorithm

The optimization function as given above, measures the value of the function, based on the input parameters, the cloud user to be selected, cloud users servicing time, total job arrival rate, total cloud servers service rate and so on. The scheduler then selects the cloud server to execute the service based on the results of the optimization function which in turn reduces the average task waiting time of each cloud users in an efficient manner. As a result of queue and load-based optimization algorithm, the rate of throughput is improved.

IV. EXPERIMENTS AND EVALUATIONS

In this section, we present an evaluation for our queue-based load optimization algorithm in terms of performance with respect to certain performance metrics, throughput, average task waiting time, memory utilization with respect to the job assigned, and number of cloud centers respectively. The proposed QS-LO model is used in Amazon Access Samples dataset and Landsat 8 data on AWS to check the performance of the proposed algorithm in a larger environment

A. Datasets and Parameter Settings

CloudSim [21], a tool for emulating a cloud computing environment is used in this study to solve the parallel job scheduling problem. Performance analysis was conducted on a PC with 2.67 GHz Intel i7-920 CPU and 4 GB of memory running with Linux 2.6.31 and using CloudSim to construct four virtual machines in a data center. The data sets, Amazon Access Sample dataset and Landsat 8 data on AWS are employed to compare the performance of the proposed algorithm and compared with other parallel job scheduling algorithms. More precisely, Amazon Access Sample dataset includes four categories of attributes including Person_Attribute, Resource_ID, Group_ID and System_Support_ID whereas Landsat 8 data on AWS that anyone can use the on-demand computing resources to perform analysis and create new products without needing to worry about the cost of storing Landsat data. Each simulation carried 30 runs..

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

B. Simulation Results of Throughput

To evaluate the performance of QS-LO model for parallel job scheduling in multiple cloud centers, we compare it with two traditional scheduling algorithms, namely, Hyper Heuristic Scheduling Algorithm (HHSA) [1] and Global Greedy Budget and Gradual Refinement (GGB-GR) [2]. For the two datasets, access samples and lands at 8, the results in table 1 shows that the M/M/M Queuing System and Load Optimized model find better results than traditional scheduling algorithms in terms of total number of jobs assigned.

In addition, the results using Landsat 8 dataset are higher than using Access Sample dataset. Moreover, with minimum job assigned, the rate of throughput is increased, whereas with the increase in the number of jobs assigned to the cloud server, the cloud server has to process a higher number of job requests and therefore the throughput rate gets reduced with the increase in the job assigned. Moreover, the results also show that QS-LO model outperforms the other two job scheduling algorithms in cloud, namely, HHSA [1] and GGB-GR [2].

Table 1 Tabulation for throughput using access samples and landsat 8 dataset

Job assigned	Throughput (%) – using Access Samples dataset			Throughput (%) – using Landsat 8 dataset		
	QS-LO	HHSA	GGB-GR	QS-LO	HHSA	GGB-GR
5	96.50	89.19	78.37	96.14	91.06	88.32
10	93.45	86.14	75.32	95.8	90.74	87.14
15	92.89	85.86	73.83	95.14	86.21	83.83
20	89.15	84.12	71.09	93.21	84.14	81.32
25	88.45	83.42	70.39	90.25	88.19	79.13
30	84.37	81.34	69.31	86.14	83.14	75.28

The results of the throughput rate given in Fig. 4 show that the HHSA and GGB-GR get almost exactly the same throughput rate in all iterations using Landsat 8 dataset. For a lower number of jobs assigned, the higher the rate of throughput because lower the job requests for the cloud user side, the response to be provided by the cloud server is also less. Higher the job request, the rate of throughput gets reduced in all the methods. On the other hand, with a lower job assigned, rate of throughput will converge quickly.

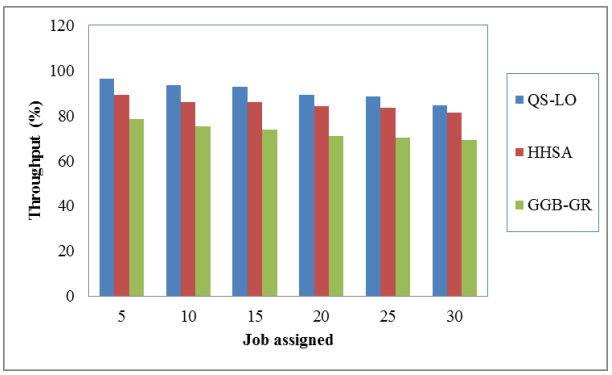

Fig. 4.a. Access samples dataset

Fig. 4 also shows that QS-LO model converges faster than the other scheduling algorithms. This is because QS-LO model is able to automatically choose the cloud server based on the job arriving requests and cloud servers service rate, and use the Poisson process to improve the results. Therefore the end results of the QS - LO model are better than the other scheduling algorithms. For example, using access sample dataset, the results depicted in Fig. 4 (a) shows that QS-LO provides a result that is close to the HHSA in terms of throughput rate. However, by applying Landsat 8 dataset, the results described in Fig. 4 (b) show that the rate of throughput is more or less similar using HHSA and GGB-GR.

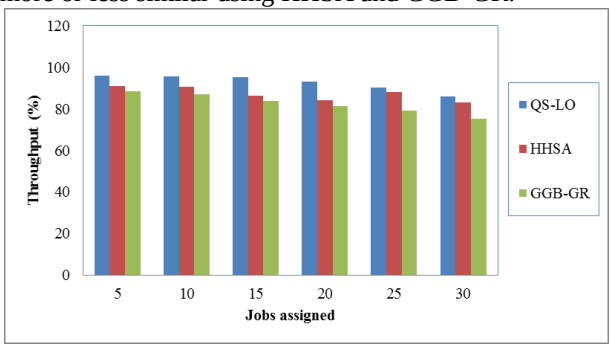

Fig. 4.b Landsat 8 dataset

Fig. 4. Convergence analysis for throughput

Fig. 4 also provides the convergence information on the scheduling algorithms compared in this study. The results given in Fig. 4 (a) show that the job request handling of these parallel job scheduling algorithms converge to a stable state very quickly when the access samples data set is applied. A good example is the results of QS-LO and HHSA, which show that the result with 5 jobs assigned is very close to the results when 20 jobs were assigned. For example, using Landsat 8 dataset, the convergence speeds of QS-LO and HHSA are quite close to each other. However, by applying access sample dataset, the convergence speeds of QS-LO and HHSA are as shown in Fig. 4 (a) and (b). They show that QS-LO has a higher chance to find a better result than HHSA [1] and GGB-GR [2] because it keeps finding a better result by applying the M/M/M Queuing System.

C. Simulation Results of Average Task Waiting Time

The average task waiting time measures the job requests made by the cloud user and the time taken by the cloud server to respond to it. Therefore the average task waiting time is mathematically evaluated as given below.

$$ATWT = \sum_{i=1}^{n} J_i * Time\ for\ each\ job \quad (9)$$

Where '$ATWT$' symbolizes the average task waiting time. The average task waiting time is measured in terms of milliseconds (ms). The average task waiting time by applying access samples and the Landsat 8 dataset for parallel job scheduling is shown in Fig. 5.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

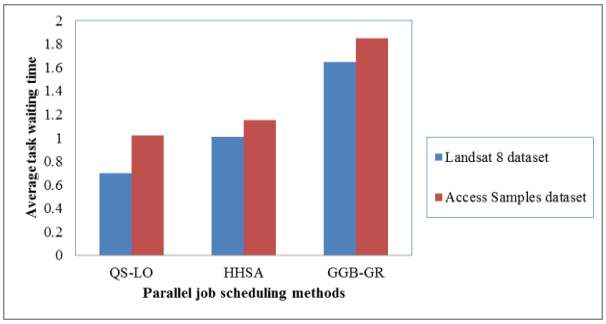

Fig. 5. Average task waiting time using Access samples and Landsat 8 dataset

The small waiting time using access sample data given in Fig. 5 implies that the load optimization function is under-loaded with jobs and thus most jobs are scheduled immediately, resulting in reducing the average task waiting time. These results also explain that for small data sets like access samples dataset that is under-loaded with job requests, the performance of the parallel job scheduling algorithms is good when compared to using large access sample dataset. On the other hand, the results of the Access sample dataset in Fig. 5 indicate that for large data sets in a queue that is fully loaded, compared to Landsat 8 dataset.

This in turn reduces the average task waiting time using QS-LO model. Moreover, the results of Fig. 5 indicate that for large datasets like access sample dataset that is fully loaded, compared to HHSA [1] and GGB-GR [2] reduces the average task waiting time of QS-LO model by 13.86% and 12.12% respectively, using Landsat 8 and Access Sample dataset. This means that by optimizing the job requests by the cloud server using optimization function, making use of the average time, utilization rate incoming jobs are scheduled sooner and therefore the average task waiting time is said to be reduced using QS-LO model when compared to the state-of-the-art methods.

D. Simulation Results of Memory Rate

Here we analyze the memory requirements and performance with respect to the job assigned to the cloud server in a parallel fashion. The memory rate is the memory required for scheduling each job and the total number of jobs assigned in the multi cloud center. The memory rate is measured in terms of kilobytes (KB) and is formulated as given below.

$$M = \sum_{i=1}^{K} J_i * \ Memory_i \qquad (10)$$

Where 'M' refers to the memory rate measured using the 'J_i' job assigned, with respect to memory consumption '$Memory_i$' for scheduling each job in multi cloud center. Lower memory rate proves the efficiency of the method.

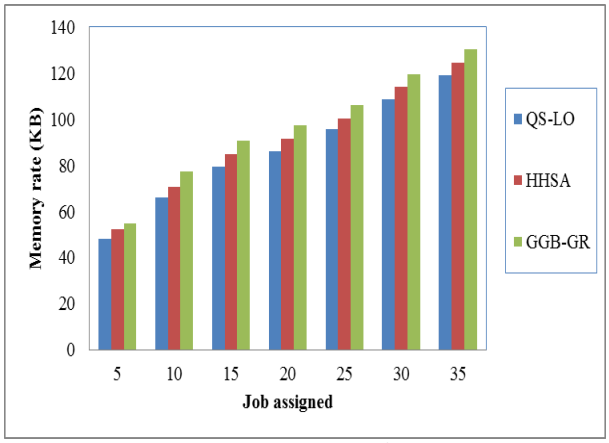

Fig. 6 Simulation results of memory rate

Fig. 6 shows the results of simulation analysis made for memory rate with respect to the incoming requests from different cloud users using Landsat 8 dataset. Executing multiple jobs using load optimized function provides significant performance gain in average time and job utilization rate using the load-based optimization algorithm. The load-based optimization algorithm is a method of scheduling, in which the user request jobs are assigned to the cloud servers according to independent and dependent jobs. The resource manager in turns assigns the virtual machines in the corresponding data cent to adapt the user's jobs. The algorithm is based on the first come, first serve for the cloud server selection process. The cloud user sends the parallel request to the cloud environment. Once the data center analyses the job and allocates the job to the first free server, the load is optimized which in turn reduces the memory required for scheduling each job.

V CONCLUSION

This paper presents an efficient solution for allocating parallel jobs in multiple cloud centers by handling M/M/M Queuing System and Load Optimized (QS-LO) model. With our proposed model, it can make the allocation of multiple jobs in a parallel fashion without collision. By using M/M/M Queuing System the cloud server have a better consolidation by analyzing the job arrival rate with a Poisson process and the job scheduler gathers the resource information from the resource manager and reduces the average task waiting time. Load-based Optimization Function optimizes the performance of multiple job requests in a cloud computing environment. The experiment result shows that the QS-LO model provides better performance with the improvement of throughput rate and reduced average task waiting time and memory rate of cloud users' requests when compared to the state-of-the-art methods.

REFERENCES

[1] Churn-Wei Tsai, Wei-Cheng Huang, Meng-Hsiu Chiang, Ming-Chao Chiang, and Chu-Sing Yang," A Hyper-Heuristic Scheduling Algorithm for Cloud", IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL 2, NO 2, APRIL-JUNE 2014

[2] Yang Wang and Wei Shi," Budget-Driven Scheduling Algorithms for Batches of MapReduce Jobs in Heterogeneous Clouds", IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL 2, NO 3, JULY-SEPTEMBER 2014

[3] Chien-An Chen, Myounggyu Won, RaduStoleru, and Geoffrey G. Xie," Energy-Efficient Fault-Tolerant Data Storage and Processing in Mobile Cloud", IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 3, NO. 1, JANUARY-MARCH 2015

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

[4] Olivier Beaumont, Lionel Eyraud-Dubois and HejerRejeb," Heterogeneous Resource Allocation underdog Constraints", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, Vol. 24, Issue 5, Jun 2012

[5] Junwei Cao, Keqin Li, andIvanStojmenovic," Optimal Power Allocation and Load Distribution for Multiple Heterogeneous Multicore Server Processors across Clouds and Data Centers", IEEE TRANSACTIONS ON COMPUTERS, VOL 63, NO 1, JANUARY 2014

[6] Junwei Cao, Kai Hwang, Keqin Li, and Albert Y. Zomaya," Optimal Multiserver Configuration for Profit Maximization in Cloud Computing", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 6, JUNE 2013

[7] Marco Polverini, Antonio Cianfrani, ShaoleiRen, and Athanasios V. Vasilakos," Thermal-Aware Scheduling of Batch Jobs in Geographically Distributed Data Centers", IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 2, NO. 1, JANUARY-MARCH 2014

[8] Sanjaya Kumar Panda, Benazir Nehab, Sujaya Kumar Sathuac," An Uncertainty-Based Task Scheduling for Heterogeneous Multi-Cloud Systems", International Journal of Information Processing, 9 (2), 13-24, 2015

[9] ShengjunXue, Managing Li, XiaolongXu, and Jingyi Chen," An ACO-LB Algorithm for Task Scheduling in the Cloud Environment", JOURNAL OF SOFTWARE, VOL 9, NO 2, FEBRUARY 2014

[10] Jiayin Li, MeikangQiu, Zhong Ming, Gang Quan, Xiao Qin, ZonghuaGu,"Online optimization for scheduling preemptable tasks on IaaS cloud systems", Journal of Parallel Distributed Computing, Elsevier, Feb 2012

[11] I. M. MaywishRajakumari, Mrs. R. Narayani,"Provenance based Adaptive Heuristic Scheduling in Cloud Environment", International Journal of Scientific & Engineering Research, Volume 6, Issue 4, April-2015

[12] LipsaTripathy, RasmiRanjanPatra," SCHEDULING IN CLOUD COMPUTING", International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol. 4, No. 5, October 2014

[13] Amit Kawalia, Susanne Motameny, Stephan Wonczak, HolgerThiele, LechNieroda, KamelJabbari, Stefan Borowski, Vishal Sinha, WilfriedGunia, Ulrich Lang, Viktor Achter, Peter Nürnberg," Leveraging the Power of High Performance Computing for Next Generation Sequencing Data Analysis: Tricks and Twists from a High Throughput Exam Workflow", PLOS ONE | DOI: 10.1371/journal. pone. 0126321 May 5, 2015

[14] Muhammad Idris, ShujaatHussain, Muhammad Hameed Siddiqi, Waseem Hassan, Hafiz Syed Muhammad Bilal, Sungyoung Lee," MRPack: Multi-Algorithm Execution Using Compute-Intensive Approach in MapReduce", PLOS ONE | DOI: 10.1371/journal. pone. 0136259 August 25, 2015

[15] Amin Nazareth, GH Dastghaibifard," Efficient Nash Equilibrium Resource Allocation Based on Game Theory Mechanism in Cloud Computing by Using Auction", PLOS ONE | DOI: 10.1371/journal. pone. 0138424 October 2, 2015

[16] Young Choon Lee, Hyuck Han and Albert Y. Zomaya," On Resource Efficiency of Workflow Schedules", 14th International Conference on Computational Science, Elsevier, Volume 29, 2014, Pages 534-545

[17] Yalda Aryan and ArashGhorbanniaDelavar," A BI-OBJECTIVE WORKFLOW APPLICATION SCHEDULING IN CLOUD COMPUTING SYSTEMS", International Journal on Integrating Technology in Education (IJITE) Vol.3, No.2, June 2014

[18] Yanyan Dai and Xiangli Zhang," A Synthesized Heuristic Task Scheduling Algorithm", Hindawi Publishing Corporation, The Scientific World Journal Volume 2014

[19] Pham Phuoc Hung and Eui-Nam Huh," An Adaptive Procedure for Task Scheduling Optimization in Mobile Cloud Computing", Hindawi Publishing CorporationMathematical Problems in Engineering Volume 2015

[20] Xiaocheng Liu, Bin Chen, XiaogangQiu, Ying CAI, and Kedi Huang," Scheduling Parallel Jobs Using Migration and Consolidation in the Cloud", Hindawi Publishing CorporationMathematical Problems in Engineering Volume 2012

[21] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Softw.: Practice Experience, vol. 41, no. 1, pp. 23-50, 2011

[22] Wei-Jen Wang, Yue-Shan Chang, Win-Tsung Lo, Yi-Kang Lee, "Adaptive scheduling for parallel tasks to QoS satisfaction for hybrid cloud environments", The Journal of Supercomputing, Springer, November 2013, Volume 66, Issue 2, pp 783-811

[23] RathnakarAchary, V. Vityanathan, Pethur Raj, S. Nagarajan, "Dynamic Job Scheduling Using Ant Colony Optimization for Mobile Cloud Computing"Intelligent Distributed Computing, Springer, Volume 321 of the series Advances in Intelligent Systems and Computing, pp 71-82, 2015.

[24] Saeed Javanmardi, Mohammad Shojafar, DaniloAmendola, Nicola Cordeschi, Hongbo Liu, Ajith Abraham, "Hybrid Job Scheduling Algorithm for Cloud Computing Environment, " Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014, Springer, Volume 303 of the series Advances in Intelligent Systems and Computing, pp 43-52, 2014