AUSTRALIAN JOURNAL OF BASIC AND APPLIED SCIENCES

ISSN:1991-8178 EISSN: 2309-8414
Journal home page: www.ajbasweb.com

Content-based Federated Job Scheduling algorithm in Cloud Computing

Dinesh Komarasamy and Vijayalakshmi Muthuswamy

Department of Information Science and Technology Anna University,Chennai, India

Address For Correspondence:
Dinesh Komarasamy, Department of Information Science and Technology Anna University,Chennai, India.
E-mail: dinesh@auist.net

ARTICLE INFO

Article history:
Received 04 December 2015
Accepted 22 January 2016
Available online 14 February 2016

Keywords:
Cloud Computing; Job Scheduling; EDF algorithm; Resource Utilization

ABSTRACT

The resource utilization of the data center has become a big hurdle of commercial cloud service providers due to the rapid expansion and variation of incoming jobs in cloud computing. In the cloud, the end user delivers both deadline and non-deadline based jobs. But, most of the existing scheduling algorithms have autonomously scheduled the deadline and non-deadline based jobs that affected the resource utilization and also violated the SLA policy in terms of deadline. So, the ultimate objective of the proposed work is to execute the deadline and non-deadline based jobs concurrently in a VM. In order to minimize the SLA violations and to improve the resource utilization, this paper proposes a new scheduling technique called Content-based Federated Job Scheduling (CFJS) algorithm in the cloud computing that will deploy in the two-tier VM architecture. In CFJS algorithm, the deadline based jobs are preprocessed based on different job constraints. After preprocessing, the deadline and non-deadline based jobs collect in the correlated job scheduler that are prioritized using the Shortest Deviation First (SDF) method. The prioritized jobs bind with the foreground VM and background VM in the VM that exists in the service provider. These contributions will mitigate the waiting time of the job, increase the resource utilization and avoid starvation. The results are simulated using a Cloudsim toolkit that shows the proposed CFJS algorithm has outperformed the other existing algorithms.

INTRODUCTION

Due to the development of several advanced network technologies, Cloud Computing is realized as an evolving computing paradigm for providing the computing resource via the Internet (Palanisamy, B., 2015). The consumers are moving to the cloud computing because of the following benefits such as On-demand service provisioning, pay per usage model or measured service, broad network access, the rapid elasticity of the resources, ubiquitous network access and location independent (Mell, P., T. Grance, 2011). Everything in cloud computing has been rendered as a service namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) (Dinesh, K., 2012). The IaaS majorly provide two different services such as compute resource and storage resources. Similarly, PaaS bestows a platform for deploying various applications or software that is provided as SaaS. Here, there are several service providers who own and control the centralized computing and storage resources. The centralized computing resources are accessed by the end-users (Dikaiakos, M.D., 2009).

The networks of physical computing nodes or physical machines are connected together for providing the vast computing and storage resources (Lee, Y.,). Due to the vast computing power existing in the cloud computing, the growing number of companies have to process the enormous number of compute-intensive jobs or data-intensive jobs (Pu, X., 2013). Here, the compute-intensive or data-intensive jobs are processed in the virtualized cloud computing environment. The virtualized cloud computing provides the VM with a particular computing and storage by the introduction of the virtualization technique. The virtualization technology permits

Open Access Journal
Published BY AENSI Publication
© 2016 AENSI Publisher All rights reserved
This work is licensed under the Creative Commons Attribution International License (CC BY).
http://creativecommons.org/licenses/by/4.0/
Open Access

To Cite This Article: Dinesh Komarasamy and Vijayalakshmi Muthuswamy., Content-based Federated Job Scheduling algorithm in Cloud Computing. ***Aust. J. Basic & Appl. Sci.,*** *10(2): 52-59, 2016*

multiple virtual machines to run on a single physical machine that operated either in the same operating system or different operating system (Barbarossa, S., 2014). The end-users approach the computing nodes existing in the cloud due to the following benefits of the virtualization technique as isolated resource provisioning, hardware independence and hiding the computing complexity of the hardware. Further, the cloud computing provides the service with high availability and reliability in a secured manner (Li, C., 2012).

Nowadays, there is an increasing interest in processing the scientific computing jobs as essentially every field is raising the demand for computational jobs and so an ample number of jobs are moving towards the cloud computing. The VMs existing in the resource pool of cloud computing have the capabilities to process the jobs efficiently. Therefore, a good job scheduling algorithm is needed to minimize the waiting time and to improve the utilization of the resource. Though there exist several scheduling algorithms to run deadline based jobs and non-deadline based jobs autonomously, none of the existing algorithms mutually process the deadline and non-deadline based jobs. Hence, this work proposes a new scheduling technique called Content-based Federated Job Scheduling (CFJS) algorithm in cloud computing to process the deadline based jobs as well as non-deadline based jobs together. In the proposed work, the deadline based jobs preprocessed with different job constraints. After preprocessing, the jobs are collected in the correlated job scheduler. The jobs in the correlated job scheduler are prioritized using the Shortest Deviation First (SDF) method. Subsequently, the prioritized jobs bind with the VM existing in the resource pool effectively. The Virtual machine monitor maintains the current information of the VM.

The rest of the paper is organized as follows. The subsequent section describes the related work. Section 3 explains the problem definition of the proposed work. Section 4 describes the design of the proposed work with the system model, job model with the effective scheduling algorithm. Section 5 describes the experimental setup with the analysis of results. The conclusion and future work of the proposed work is described in the last section.

Literature Review:

This section describes the various existing scheduling algorithms to process the jobs conventionally. At any arbitrary time, the multiple jobs are handed over to the cloud computing. Initially, the jobs are processed based on the arrival order as termed as First Come First Serve (FCFS) algorithm. But, the FCFS model cannot optimally utilize the resources of the service providers. This section explains the various existing scheduling algorithms to process the deadline based jobs and non-deadline based jobs independently (Silberschatz, A., 2011).

Deadline based job scheduling algorithm:

The deadline based job scheduling is a type of priority scheduling algorithm. Based on the characteristics of the jobs, the deadline based jobs are categorized as preemptive and non-preemptive jobs (Badgujar, S.Y., A. Bone, 2014). Though the non-preemptive job cannot stop its execution once it starts its execution, the preemptive jobs can swap the executing job with the job at the head of the queue whenever the high priority jobs arrive compare than the running job. Several existing algorithms prioritized the jobs in different ways. Different scheduling algorithms were introduced to schedule the jobs in homogeneous and heterogeneous environment. Initially, the Earliest Deadline First (EDF) algorithm prioritized the jobs based on their deadline. The EDF algorithm supported the non-preemptive job scheduling called non-preemptive EDF algorithm (np-EDF) (Van den Bossche, R., 2013). The jobs are arriving continuously in the cloud system. So, the high priority jobs arrive during the execution of the job and so the processing job is preempted to process the high priority job. So, the EDF algorithm has been modified to support the preemptive EDF scheduling algorithm called a fully preemptive EDF algorithm (fp-EDF). The context switching needs some processing speed of the resources during the preemption of the jobs. Therefore, the jobs were preempted based on the set of conditions called as controlled preemptive EDF algorithm (cp-EDF) to improve the utilization of the resources (Jinkyu, 2014).

In addition to the job deadline, the deadline based jobs were prioritized by considering the some other attributes of the job as the length of the job. Thus, the Minimum Variation First (MVF) algorithm has been proposed to prioritize the jobs using the different attributes such as deadline of the job, the length of the job and processing speed of the VM. Further, the MVF algorithm was modified as improved MVF (iMVF) algorithm by recognizing the arrival time of the job to avoid starvation (Dinesh Komarasamy, Vijayalakshmi Muthuswamy, 2014). Further, the jobs were optimally scheduled using Partial Critical Path (PCP) algorithm in order to complete the jobs within the deadline. In PCP algorithm, the jobs were completed with the support of deadline distribution and planning phase (Abrishami, S., 2013). The PCP was modified as Enhanced Infrastructure Cloud PCP with Replication (EIPR) (Calheiros, R.N., R. Buyya, 2014).

Moreover, the jobs were completed by running several processor to complete within the deadline. But, job replication raised the budget of the running jobs (Plankensteiner, K., R. Prodan, 2012). So, the replication impact played a significant factor to a tradeoff between the job replication and budget (Rodriguez, M.A., R. Buyya, 2014). The above algorithms were not processed the jobs efficiently in a heterogeneous environment.

Here, the jobs were allocated to the VM by computing the minimum required processing speed of the job. After computing the minimum processing speed of the job, the jobs mapped with the VM which have the capabilities to process the job in a cost-effective manner (Huang, D., 2014). Further, the jobs processed by combining based on the in-stage local greedy algorithm and dynamic provisioning. Here, two greedy algorithms were introduced to minimize the job execution time as the global greedy Budget (GGB) and gradual Refinement (GR) (Wang, Y., W. Shi, 2014).

Non-deadline based job scheduling algorithm:

Unlike the above deadline based scheduling algorithms, Like time-sharing system, the jobs were processed using the Round Robin (RR) algorithm (Alnowiser, A., 2014). Further, the jobs were scheduled using the Shortest Job First (SJF) in terms of the job length. The jobs that need multiple machines for processing, scheduled based on the FCFS algorithm. Moreover, the EASY backfilling algorithm was introduced by changing the order of job execution whenever the available VMs were inadequate to process the job at the head of the queue to improve the processor utilization (Liu, X., 2013). The non-deadline based jobs processed depending on the behavior of the underlying resources termed as a resource-aware scheduling. The blind scheduling algorithms were used for processing the non-deadline based jobs. Here, the non-deadline based jobs were forwarded to the data center which have the idle computing power called as Blind Online Scheduling Algorithm (BOSA). The BOSA scheduled the jobs without knowing the arrival rate and processing rate of the job (Zhou, L., H. Wang, 2013).

Problem Defnition:

A large number of jobs are submitted to the cloud system by the end user with different content as deadline based and non-deadline based jobs. From the state of the art, several existing scheduling algorithms scheduled the deadline based jobs and non-deadline based jobs independently. Here, the submitted job is considered as a compute-intensive job that need utmost of the processing time for executing the job. So, the computing capacity of the VM partitioned into foreground and background VM in the two-tier VM architecture. In the two-tier architecture, the deadline based jobs cannot process in the foreground as well as in background VM because it is very difficult to predict expected execution time of the job. Moreover, the non-deadline based jobs processed in the VM based on the resource-aware scheduling algorithm. To our best knowledge, none of the existing scheduling algorithms federally process deadline based jobs and non-deadline based jobs. Thus, this paper introduces Content-based Federated Job Scheduling (CFJS) algorithm in cloud computing that will process the deadline based jobs and non-deadline based jobs concurrently in a two-tier VM architecture. Here, the deadline based jobs process in the foreground VM and non-deadline based jobs process in the background VM to improve the resource utilization and meet the SLA requirements.

Design Of Proposed System:

In the proposed system, the web interface pulls together the numerous clients request in the cloud system. Fig 1 explains the design of the proposed system. In the proposed work, the cloud system model is considered as a centralized scheduling model. The jobs are scheduled act as a master and slave model. Here, the proposed algorithm acts master and the allocation of job with the respective VM is considered as a slave. The cloud system composed of several service providers $C_s = \{p_1, p_2, ..., p_n\}$ where C_s represents the cloud service provider and p_i represents the service provider in cloud system. For simplicity and better understanding, the system model considers only single service provider to deploy the scheduling algorithm. Similarly, the job model is assumed in the reverse model of master and slave. Here, the various clients are considered as slaves and the scheduling algorithm is taken as a master that will process the jobs using the VMs in service provider. The jobs are considered as nonpreemptive based on the behavior of the job. Moreover, the jobs are aperiodic in nature which means the arrival time of the job are not known earlier in this work.

In the proposed work, the web interface puts forward the jobs in the cloud system $J = \{j_1, j_2, ..., j_n\}$ where n represents the total number of jobs in the job queue. During the job submission, the user should provide some additional information such as a length of the job and the arrival time of the job $j_i = \{ls_i, a_i\}$ but the deadline of the job is considered as optional d_i. The incoming jobs arrive at a constant arrival rate is assumed in the proposed work. Initially, the jobs are stored in the job queue that is termed as j_q. After submitting the jobs, the jobs are handed to job classifier. The job classifier embeds with a two queue to store the deadline based and non-deadline based jobs independently. The job classifier categories the incoming jobs as deadline and non-deadline based jobs using the information provided by the end-user. The job classifier is represented as J_c and categorized as given below.

$$J_c = \begin{cases} j_i \in j_d & ;if\ j_i haved_i\ \forall_j in\ j_q \\ j_i \in j_{nd} & ;otherwise \end{cases}$$

where j_d *and* j_{nd} represents the queue having the deadline and non-deadline based jobs respectively. Here n_1 and n_2 jobs come under deadline based jobs and non-deadline based jobs respectively (i.e. $n = n_1 + n_2$). After classification, the deadline based jobs are forwarded for preprocessing in order to avoid the unwanted delay. The jobs are preprocessed by comparing the maximum processing speed of the job with the minimum required processing speed of the job. The minimum required processing speed of the job is represented as p_s and computed as given below.

$$p_{si} = \frac{ls_i}{d_i - a_i} ; \forall_j \, in \, j_d \; where \; j \in (1, n_1)$$

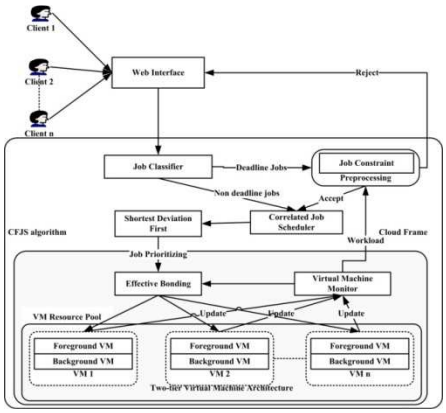

Fig. 1: Proposed System Design.

The maximum processing speed of the VM existing in the service provider is represented as $\max_s$ and it is computed as given below.

$$\max_s = \max(vmp_{si}) \; where \; i \in (1, m)$$

Where m and vmp_s represents the number of VM in the service provider and processing speed of the VM existing in the service provider of the cloud system. The VMs information are available in the VM monitor. After p_s and $\max_s$, the jobs are preprocessed by comparing p_s and $\max_s$ as given below.

$$p_{pj} = \begin{cases} 1 & ; if \; p_s \le \max_s \\ 0 & ; otherwise \end{cases} \quad \forall_j \in j_d$$

where p_p represents the preprocessed jobs. The preprocessed jobs are stored in the j_{dd}. After preprocessing, the jobs are moving to the correlated job scheduler. Afterwards, the correlated job scheduler coalescing both deadline and non-deadline based jobs. Later on, the jobs forward to the Shortest Deviation First (SDF) method. The SDF approach prioritizes the jobs by computing the deviation value of processing speed. The deviation value is represented as d_v and computed by comparing p_s and $\max_s$ and computed as given below.

$$d_{vj} = \max_{sj} - p_{sj} \quad ; where \; \forall_j \, in \, j_{dd}$$

After computing the deviation value, the job which is having the least value is assigned as a higher priority value. On the other hand, the non-deadline based are also prioritized using the ls with the vmp_s. The priority of the non-deadline based jobs is represented as p_{nd} and computed as given below

$$p_{nd} = \frac{vmp_{sj}}{ls_j} + a_j \quad ; \forall_j \, in \, j_{nd}$$

After prioritizing, the job which is having the largest value is assigned as a high priority job. After prioritizing, the deadline based jobs and non-deadline based move forward to the effective bonding. The effective bonding will map the job with the VM for processing the jobs effectively. During the execution of the job, the job cannot utilize the full processing speed of the VM. Hence, the computing capacity of the VM is partitioned as foreground and background VM that will dynamically share the processing speed of the VM. The foreground VM process the deadline based jobs and the background VM process the non-deadline based jobs

until the jobs in $j_{dd} \vee j_{nd} = \phi$. The effective bonding algorithm binds the jobs with the suitable VM as given below

$$E_b = \begin{cases} \min\left(\frac{p_{sj}}{vmp_{sk}}\right) & ;\forall_k \forall_j \, in \; j_{dd} \\ \max\left(\frac{ls_j}{vmp_{sk}}\right) & ;\forall_k \forall_j \, in \; j_{nd} \end{cases} \quad where \, k(1,m)$$

where m represents the number of VM existing in the VM monitor. The deadline based jobs are mapped with the suitable VM because the subsequent jobs may need more processing speed. So, the jobs are bound with the most suitable in order to minimize the SLA (Service Level Agreement) violation. The non-deadline based jobs are effectively processed in the background VM. Moreover, the deadline based jobs may also process in the background VM whenever the $j \; in \; j_{dd} \leq m$ and vice versa. The jobs are effectively bonding with the VM due to support of VM monitor. The VM monitor maintains the present status of the foreground VM and background VM.

CFJS Algorithm in cloud computing:

(1) Begin

(2) $j_q \leftarrow$ holds the list of incoming jobs

(3) The j in j_q categorized into $j_d \; \& \; j_{nd}$ as

$$J_c = \begin{cases} j_i \in j_d & ;if \; j_i have \, d_i \\ j_i \in j_{nd} & ;otherwise \end{cases} \forall_j in \; j_q$$

(4) The j in j_d preprocessed by computing $p_s \; \& \max_s$

$$p_{si} = \frac{ls_i}{d_i - a_i} ; \forall_j \; in \; j_d \; where \; j \in (1, n_1)$$ store in j_{dd}

$$\max_s = \max(vmp_{si}) \; where \, i \in (1,m)$$

$$p_{pj} = \begin{cases} 1 & ;if \; p_s \leq \max_s \\ 0 & ;otherwise \end{cases} \quad \forall_j \in j_d$$

(5) After preprocessing, the jobs in $j_{dd} \; \& \; j_{nd}$ prioritizes

$$d_{vj} = \max_{sj} - p_{sj} \quad ; where \; \forall_j \; in \; j_{dd}$$

$$p_{nd} = \frac{vmp_{sj}}{ls_j} + a_j \quad ; \forall_j \; in \; j_{nd}$$

(6) After prioritizing, map the job with VM

$$E_b = \begin{cases} \min\left(\frac{p_{sj}}{vmp_{sk}}\right) & ;\forall_k \forall_j \, in \, j_{dd} \\ \max\left(\frac{ls_j}{vmp_{sk}}\right) & ;\forall_k \forall_j \, in \, j_{nd} \end{cases} \quad where \, k(1,m)$$

(7) if $(j_i \; in \; j_{dd} \; and \; j_i \; in \; j_{nd}) \geq m$ then \\predict expected processing speed is difficult

allocate $j_i \; in \; j_{dd} \rightarrow foreground \, vm$

allocate $j_i \; in \; j_{nd} \rightarrow background \, vm$

(8) Elseif $j_i \; in \; j_{dd} < m \; and \; j_i \; in \; j_{nd} \geq m$ then \\ more number of non-deadline based job

allocate $j_i \; in \; j_{dd} \rightarrow foreground \, vm$ and assign

$j_i \; in \; j_{nd} \rightarrow both \; foreground \& background vm$

(9) Else \\ to improve resources utilizationassign $j_i \; in \; j_{dd} \rightarrow foreground \, \& \, background \, vm$

$j_i \; in \; j_{nd} \rightarrow both \; foreground vm$

(10) End if

(11) End if

(12) End

The above CFJS algorithm simultaneously process the deadline base jobs and non-deadline based jobs with the support of two-tier VM architecture. The CFJS algorithm is proposed to minimize the SLA violation by completing the job within the deadline and increase utilization of the resources.

Experimental results and simulation results:

Cloudsim is a simulation toolkit to provide the computing cloud environment for testing and demonstrating the jobs in different specification. The cloudsim composed of several classes to develop a testing platform. The cloudsim contains a VM, host and data centers. The table 1 describes the different simulation parameters to analysis the performance of the proposed system

Table 1:.Simulation Parameters.

Parameter	Range
Job type	Compute-intensive job
Job size (MI)	10,000-20,000
Number of jobs	10-50
Computing power of Data Center (MIPS)	5750
Deadline (MS)	15-75

With the support of the above simulation parameters, the jobs are effectively processed in the VM existing in the data center.

Waiting time of the job:

The proposed CFJS algorithm minimizes the waiting time of the job by utilizing the remaining idle computing power of the VM. The remaining computing power of the VM is utilized by run non-deadline based jobs in the VM. The CFJS algorithm deployed in the two-tier VM architecture by running the deadline based jobs and non-deadline based jobs using the foreground and background VM respectively. Fig 2 represents the waiting time of the job. The deadline based jobs are processed in the foreground VM. The CFJS algorithm optimally utilizes the processing speed of the VM that reduces the waiting time of the job when the number of jobs is greater that number of VM. The CFJS algorithm outperforms the other existing algorithm like FCFS and cloudsim by concurrently running the jobs in the VM.

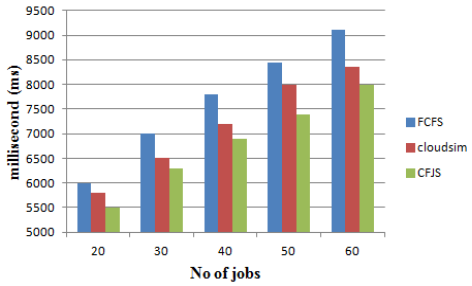

Fig. 2: Waiting time of the job.

Resource Utilization:

The proposed CFJS algorithm utilizes the computing power of the VM effectively by running the deadline based and non-deadline based jobs concurrently in the foreground and background VM. Fig 3 shows the resource utilization using CFJS algorithm.

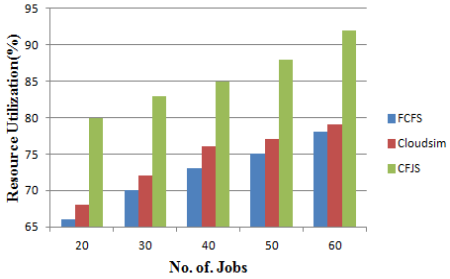

Fig. 3: Resource Utilization.

The jobs are assumed as a nonpreemptive and so reduces the number of context switching that will avoid the wastage of computing power of the VM. The jobs effectively utilize the resources whenever the number of incoming jobs is greater than the available number VM in the proposed system. The CFJS algorithm outperforms the other existing algorithms like cloudsim and FCFS by running the jobs simultaneously in the foreground and background VM.

Conclusion and Future work:

Several algorithms have been developed to process the jobs among the resources exists in the cloud system. The existing algorithm separately executes the deadline and non-deadline based jobs that affect the utilization of the resource and SLA violating deadline based jobs. Thus, this paper proposed a new scheduling technique called Content-based Federated Job Scheduling (CFJS) algorithm that deployed in the two-tier VM to collectively process deadline based jobs and non-deadline based jobs. Based on the end user information, the job classifier classifies the deadline and non-deadline based jobs in the proposed system. The number of SLA violations was reduced by preprocessing and prioritizing the deadline based jobs that avoids the unwanted delay. Afterwards, the jobs bind with the appropriate VM that minimizes the waiting time of the job. The idle computing power of the VM is utilized to process the non-deadline based jobs to improve the utilization of the resources. The proposed CFJS algorithm outperforms the other existing algorithm by running the deadline based jobs along with the non-deadline based jobs in a particular VM that minimizes the waiting time of the job and maximizes the throughput of the system. In the future, an energy efficient scheduling algorithm can be developed for the cloud system.

REFERENCES

Abrishami, S., M. Naghibzadeh, D.H. Epema, 2013. Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds, Future Generation Computer Systems, 29(1): 158-169.

Alnowiser, A., E. Aldhahri, A. Alahmadi, M.M. Zhu, 2014. Enhanced weighted round robin (ewrr) with dvfs technology in cloud energy-aware, IEEE International Conference on Computational Science and Computational Intelligence (CSCI), 320-326.

Badgujar, S.Y., A. Bone, 2014. Cloud resource allocation as preemptive scheduling approach, International Journal of Computer Applications, 88(18).

Barbarossa, S., S. Sardellitti, P. Di Lorenzo, 2014. Communicating While Computing: Distributed mobile cloud computing over 5G heterogeneous networks, IEEE in Signal Processing Magazine, 31(6): 45-55.

Calheiros, R.N., R. Buyya, 2014. Meeting deadlines of scientific workflows in public clouds with tasks replication, IEEE Transaction on Parallel and Distributed Systems, 25(7): 1787-1796.

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A. De Rose, R. Buyya, 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 41(1): 23-50.

Dave, Y.P., A.S. Shelat, D.S. Patel, R.H. Jhaveri, 2014. Various job scheduling algorithms in cloud computing: A survey, International conference on Information Communication and Embedded Systems (ICICES), 1-5.

Dikaiakos, M.D., D. Katsaros, P. Mehra, G. Pallis, A. Vakali, 2009, Cloud Computing: Distributed Internet Computing for IT and Scientific Research, IEEE Internet Computing, 13(5): 10-13.

Dinesh Komarasamy, Vijayalakshmi Muthuswamy, 2014. Job scheduling using Minimum Variation First algorithm in cloud computing, Sixth International Conference on Advanced Computing (ICoAC), 195-198.

Dinesh, K., G. Poornima, K. Kiruthika, 2012, Efficient Resources Allocation for different Jobs, International Journal of Computer Application, 56(10): 30-35.

Huang, D., C. Zhu, H. Zhang, X. Liu, 2014. Resource intensity aware job scheduling in a distributed cloud, China Communications, 11(14): 175-184.

Jinkyu, Lee. Shin, K.G., 2014. Preempt a Job or Not in EDF Scheduling of Uniprocessor Systems, IEEE Transactions on Computers, 63(5): 1197-1206.

Lee, Y., S. Len, R. Chang, Improving job Scheduling algorithm in a grid environment, Future Generation Computer Systems, 27(8): 991-998.

Li, C., A. Raghunathan, N.K. Jha, 2012. A Trusted Virtual Machine in an Untrusted Management Environment, IEEE Transactions on Services Computing, 5(4): 472-483.

Liu, X., C. Wang, B.B. Zhou, J. Chen, T. Yang, A.Y. Zomaya, 2013. Priority-based consolidation of parallel workloads in the cloud, IEEE Transactions on Parallel and Distributed Systems, 24(9): 1874-1883.

Mell, P., T. Grance, 2011. The NIST Definition of Cloud Computing: NIST Special publication, 800-145.

Palanisamy, B., A. Singh, Ling Liu, 2015. Cost-Effective Resource Provisioning for MapReduce in a Cloud: IEEE Transactions on Parallel and Distributed Systems, 26(5): 1265-1279.

Plankensteiner, K., R. Prodan, 2012, Meeting soft deadlines in scientific workflows using resubmission impact, IEEE Transactions on Parallel and Distributed Systems, 23(5): 890-901.

Pu, X., L. Liu, Y. Mei, S. Sivathanu, Y. Koh, C. Pu, Y. Cao, 2013. Who Is Your Neighbor: Net I/O Performance Interference in Virtualized Clouds, IEEE Transactions on Services Computing, 6(3): 314-329.

Rodriguez, M.A., R. Buyya, 2014. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds, IEEE Transactions on Cloud Computing, 2(2): 222-235.

Silberschatz, A., P.B. Galvin, G. Gagne, 2011. Operating System Concepts, ISSN- 978-1-118-06333-0, 9th edition, John Wiley & Sons.

Van den Bossche, R., K. Vanmechelen, J. Broeckhove, 2013. Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds, Future Generation Computer Systems, 29(4): 973-985.

Wang, Y., W. Shi, 2014. Budget-driven scheduling algorithms for batches of MapReduce jobs in heterogeneous clouds, IEEE Transaction on Cloud Computing, 2(3): 306-319.

Zhou, L., H. Wang, 2013. Toward blind scheduling in mobile media cloud: Fairness, simplicity, and asymptotic optimality, IEEE Transactions on Multimedia, 15(4): 735-746.