J. Comp. Sci. Comm., 2(1)1-10, 2016

ISSN 2356-8844
Journal of Computer Sciences and Communication
www.textroad.com

A Newly Proposed Workflow Scheduling Algorithm to Fulfill the Requirements of Most Users and Cloud Computing Environment

Minoo Soltanshahi

Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran

ABSTRACT

Cloud computing is the latest distributed technology of dynamically shared resources, which can respond to user requirements through allocation of resources to their applications. The workflow user applications refer to a set of tasks to be processed within the cloud environment. As one of the key operations in cloud computing environments greatly affecting the system performance, task scheduling involves certain algorithms. In order to achieve higher efficiency of resources in a cloud environment, this paper presented a new scheduling algorithm with better performance than its counterparts. Designed based on min-min algorithm providing an effective scheduling for curtailing the execution time of tasks, the newly proposed scheduling algorithm can appropriately respond to most requirements of users and cloud environments such as prioritizing tasks, load balancing, meeting the deadline and fault tolerance. Additionally, the algorithm proposed for calculating the cost adopts a combined strategy of SLA and QOS. According to the results obtained by simulation in CloudSim, the newly proposed algorithm has the best performance among similar algorithms, fulfilling numerous requirements and managing to curtail execution time.
KEYWORDS: scheduling algorithm, load balancing, prioritizing tasks, deadline, cost constraint, fault tolerance

1 INTRODUCTION

Cloud computing refers to development and deployment of Internet-based computer technology. In fact, it is a method of computing in a space where IT-related capabilities are offered as services to the users, enabling them to gain access to technology-based services over the Internet without any specialized knowledge about these technologies or any need to personally handle the cloud infrastructure. The structure of cloud computing resembles a massive cloud through which users anywhere in the world can gain access to resources.

Therefore, it is crucial to realize the effective utilization of these resources [1-4]. When the user submits a certain application for executing in the cloud environment, it will be processed as a workflow. Every workflow in the cloud computing environment is usually displayed by a Directed Acyclic Graph (DAG), where the tasks and their interrelationships are represented by nodes and edges, respectively.

An effective strategy on cloud performance involves scheduling algorithm whose task is to correctly map the tasks on the resources. In fact, correct mapping of tasks can leave a beneficial impact on efficiency of resources, minimization of execution time and cost and maximization of fault tolerance. There have been numerous studies conducted on this area, each proposing an algorithm meeting certain requirements such as cost, time, fault tolerance, load balancing, power consumption, etc [5-20]. However, there are few studies finding an algorithm responding ideally to the cloud environment, each coming with its own advantages and disadvantages.

Therefore, this paper intended to propose a new efficient algorithm dubbed flpmm (fault tolerance and load balancing and priority based min-min algorithm), which can be useful in improving system performance. Since the proposed algorithm takes into account the requirements of both users and cloud environments, it can provide a desirable strategy for allocation of resources. In fact, flpmm meets several requirements such as prioritizing tasks, deadlines, load balancing and fault tolerance. Moreover, it can calculate cost through a strategic combination of quality of service (QOS) reflecting the user expectation for running applications, and service-level agreement (SLA) specifying the level of service quality agreed by users and cloud providers. In this procedure, the cost constraint is evaluated upon request of the user.

In case there are no suitable resources, the payment method would be based on pre-specified periods of usages. The results of simulation in Section Four revealed that the new algorithm has better performance

*Corresponding Author: Minoo Soltanshahi, Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran. Email: Minoo.soltanshahi@gmail.com

compared to its counterparts, fulfilling most of the requirements of users and cloud environments, and yet curtailing the execution time. This paper has been segmented as follow: section two will explore the relevant literature, section three will elaborate on the proposed algorithm, and section four will present the simulation and its results. Finally, section five will discuss the conclusions.

2 RELATED WORK

Scheduling tasks on resources is a well-known NP-complete problem, which can be associated with several factors, including heterogeneous and dynamic characteristics of sources and applications within the cloud computing environment for which multiple algorithms have been designed so far [21,5].

Studies conducted by [7] indicated that min-min algorithm can provide the best scheduling for reduction of makespan compared to minimum time execution (MTE), minimum cost execution (MCE), max-min and etc.

Min-Min is initiated with a set of tasks not assigned. First of all, a minimal completion time is achieved for every task. Then, the lowest value of minimum completion time between all tasks is selected on a given resource. Accordingly, the task is scheduled on the corresponding machine. At the next stage, the execution time for all other tasks on the machine is updated by adding the execution time of the allocated task to execution time of other tasks on the machine. Then, the allocated task is removed from the list of tasks assigned to the machine. The same procedure is replicated until all tasks are allocated to the resources [18-20,22-23].

Figure1 illustrates min-min algorithm in which C_{ij} is the completion time of *i*th task on *j*th resource, calculated through the following equation (1),

$$C_{ij} = E_{ij} + r_j \quad (1)$$

Where E_{ij} represents the execution time of *i*th task on *j*th resource, while r_j represents the ready time of resource j. However, this method has a major drawback potentially leading to starvation [23].

1. ***For*** *all tasks* $\boldsymbol{t_i}$ *in* ***MT*** *(in an arbitrary order)*
2. ***For*** *all Machines* $\boldsymbol{m_j}$ *(in a fixed arbitrary order)*
3. $\boldsymbol{C_{ij} = E_{ij} + r_j}$
4. ***Do*** *until all tasks in* ***MT*** *are mapped*
5. *For each task in* ***MT*** *find the earliest completion time and the machine that obtains it.*
6. *find the task* $\boldsymbol{t_k}$ *with the* ***minimum*** *earliest completion time.*
7. *assign task* $\boldsymbol{t_k}$ *to the machine* $\boldsymbol{m_l}$ *that gives the earliest completion time.*
8. *delete task* $\boldsymbol{t_k}$ *from* ***MT***
9. *update* $\boldsymbol{r_l}$
10. *update* $\boldsymbol{C_{il}}$ *for all* $\boldsymbol{i}$
11. ***End Do.***

Fig 1. The traditional min-min scheduling algorithm [23]

The two-stage load balancing OLB+LBMM is a scheduling algorithm combining Opportunistic Load Balancing (OLB) and Load Balance Min-Min (LBMM) for achieving better system performance and load balancing. OLB maintains each node in working mode so as to achieve higher load balancing, while LBMM scheduling algorithm reduces the execution time of each task on the node, thus curtailing the total execution time. This algorithm is applied in a three-level cloud computing network where efficiency and productivity of resources are taken into account [18].

Moreover, improve min-min was proposed by Rajwinder Kaur et al. (2013), executed through the following stages. At first, min-min algorithm is executed as task T is assigned to resource R with the shortest execution time. Then, resources are arranged based on execution time, calculating the makespan and selecting a resource capable of responding to makespan. At the next stage, executed tasks will find the resources producing makespan. Then, it will find the minimum completion time of those tasks and the resources capable of responding. It will apply the settings on every single task. If the next completion time of task is shorter than makespan and the new completion time of machine is shorter than makespan, it will schedule the task on the responding resource. Finally, it will update the ready time of both resources [24].

Afterwards, algorithm LBIMM adds the capability of load balance to the optimized algorithm min-min, where the tasks are first mapped on the resources, and then the smallest tasks are selected from resources with heavy loading, calculating their completion times on the rest of resources. It then compares the shortest completion time against the output time of min-min. If it is shorter than completion time of min-min, the task will be mapped on that particular resource, updating the ready time of other resources. This process is repeated until other resources cannot produce a completion time shorter than that of the task on the heavy resource [20, 16].

In 2013, Chen et al. presented PALBIMM algorithm where prioritizing tasks was added to parameters involved in LBIMM algorithm. Fulfilling this requirement can be critical for users. This algorithm attempts to fulfill the requirements of load balancing, prioritizing tasks and minimization of execution time [16]. Later on, Mangla et al. (2015) added a new feature to PALBIMM algorithm known as recovery from failure, dubbing it RPALBIMM. In the newly proposed algorithm, effort was made to maintain the workflow whenever the service failed for any reason by sending the tasks to an appropriate service [17].

Each of the mentioned algorithms were presented to meet several requirements such as curtailing execution time, load balancing and prioritizing tasks. The new algorithm proposed in the current study has objectives similar to those of PALBIMM, but the execution policy is completely different. Moreover, the new algorithm entails a new strategy for cost payment and capability to enhance fault tolerance. The simulation results indicated that the new algorithm has desirable performance. As noted above, RPALBIMM algorithm added the capability of recovery from failure to PALBIMM, which is different from the policy of enhancing fault tolerance, because the new algorithm serves to prevent the incidence of failure through estimating the execution time.

3 PROPOSED METHOD

The efficiency of any cloud environment is dramatically affected by accurate scheduling of tasks on resources. Hence, the scheduling algorithms should be designed in a way to respond equally to requirements of users and provide beneficial utilization of the cloud resources. The new algorithm proposed in this paper can eliminate the shortcomings found in the model in which the needs of both the user and cloud environment have been considered. There are several requirements met by the proposed algorithm, such as prioritizing tasks, meeting the deadlines, adopting a new cost-calculating model combining QOS and SLA, load balancing and enhancing fault tolerance. The next section will explore the new algorithm through a string of code related to the scheduling operation and a parallel string of code related to fault tolerance.

3.1 Scheduling operation code

The user applications are received as a workflow to be executed in the cloud computing environment. Each workflow is a set of tasks sent to the scheduler to perform a given job. The scheduler will then find the appropriate resource for mapping the tasks on resources. Some of these tasks may be considered more important by the user who wishes to perform these tasks first. In the newly proposed model, the tasks are initially divided into two groups of high and low priority based on user's preferences. The scheduling operation is first conducted on the high priority group and then on the low priority group so long as there are tasks not yet assigned. After receiving and categorizing the tasks, the execution speed is increased so as to avoid idle resources. The tasks are mapped on resources through min-min algorithm. When the resources are busy executing tasks, the next input tasks will be subject to policies of deadline, cost calculation and load balancing. At the second stage, an appropriate resource is selected first by evaluating the deadline against the total completion time of tasks within the queue as well as the new task on the resource. Moreover, the execution cost and the cost constraint will be evaluated upon request of the user. If there are resources available meeting both conditions, the task will be mapped. Otherwise, the deadline is solely evaluated. If a resource is found capable of executing the task within the deadline specified by the user but failing to meet the cost constraint, the cost will be calculated based on pay-as-

you-go pricing model and the task will be mapped. In the pay-as-you-go pricing model for specified periods, the cost is usually calculated by cloud providers. One of the parameters of quality involves service-level agreement (SLA), on which many models have been developed. Some of these models entail spans as short as 5 minutes in order to achieve effective utilization of the last span. In fact, if longer spans such as 1 hour is to be considered, the user may not completely use the last span and leave a small fraction, the cost of which should be fully paid [8]. Accordingly, the 5-minute spans were recommended in the new algorithm. If the service under that deadline and cost constraint was not found, min-min algorithm will be implemented as an available service is selected capable of carrying out the job within the shortest possible time. Then, the cost will be evaluated to determine whether it is higher than the cost constraint specified by the user. If so, the execution cost will be calculated based on the time used from the requested period. Load balancing is another parameter involved in utilization of resources in large-scale distributed environments such as cloud, greatly contributing to system performance. Any failure in taking this measure into account may lead to heavy loading in certain resources and idleness in others. Nevertheless, the tasks in an active service queue may wait for an excessively long time. This can increase execution time, leading to incomplete execution of tasks within the given deadlines. Hence, failure in load balancing will not only escalate execution time and cost but also curtail the system fault tolerance. The algorithms proposed so far for that purpose have mainly involved queue length and size of tasks, based on which tasks are equally mapped on other resources. However, these criteria can hardly provide the load balancing conditions within a cloud environment. Given the example below, it will not be reasonable. Let us assume that a virtual machine with processing power of 100 MIPS entails 5 tasks with a total size of 80 MIPS. On the other hand, a second virtual machine with processing power of 20 MIPS entails 2 tasks with a total size of 15 MIPS. In this scenario, as a new task is inserted, the second machine would be selected for considering the queue length and size of tasks, which is not reasonable. The first machine, however, would be the more reasonable option since it has a higher processing power. Hence, flpmm considered the processing power of virtual machines so as to establish the load balancing conditions. In fact, the processing power for tasks in the queue is calculated through a new task, i.e. tasks will be mapped on resources meeting the required processing power. Figure 2 illustrates the newly proposed algorithm.

```
High or low priority groups ← task
Cost ← cost constraint
D  ← Deadline
N ← totall number of existing resources
S(n) ← all existing and new resources
For all task in high or low group run min-min algorithm in s(n)  and scheduling task
on feasible resource
 \* at first run into high then low group
Do while tasks groups is not empty \* at first high then low group
      For each task
      For all resources in S(n)
      Compute time and cost execution task in S(n)
      If( S (i time ) <=(Task d)) and ( S(i cost) <= (Task cost))
            Return resource and add task to ready queue resource
            Delete task from high or low group
      Else if (S (i time ) <=(Task d))
           Return resource and add task to ready queue resource
            Delete task from high or low group
           Calculate price
      Else
          Rm = run min-min algorithm in s(n)
         compute cost for execution task on Rm
          if ((Rm cost) > (Task cost))
              calculate price
          end if
     End if
End do
```

Fig 2. The proposed scheduling algorithm

3.2 code for fault tolerance

In cloud computing systems, a service may for any reason after scheduling fail to properly complete a particular task within the deadline assigned by the user. Specifically within workflows, it may last for several months and ultimately fail. In order to enhance fault tolerance and prevent any failure parallel to scheduling operation until all tasks have not been completed, the new algorithm will involve the following scenario. At first, completion time for each task under execution is estimated. Then, the estimated completion time will be compared against the deadline. If the completion time is longer than the deadline, then min-min algorithm will be implemented on all available services. The completion time of task on the output service of min-min will be compared against the time required for completing the task on the current service. Should it be shorter, the task will be mapped on that service. The results stored from execution of task will be transferred from previous service to the new service as the task execution is maintained on the new service. Otherwise, the task execution will continue on the same current service. Figure 3 illustrates the fault tolerance operation.

```
Estimate need time for task
If ( Task(time need) > Task(time deadline)
      Rm =Run min-min algorithm  in s(n)
     if (Rm (time execution) < Task(time need))
       Store  current state in storage cloud and Return
resource
       Reassign task to resource and get current state from
cloud storage
     else
       Continue run on current resource
     end if
End if
```

Fig 3. The parallel fault tolerance operation

4 RESULTS OF SIMULATION

CloudSim is a tool that supports modeling and developing one or several virtual machines on simulated server features in a data center, their functions and mapping onto appropriate virtual machines. In fact, CloudSim provides a library with numerous default classes coded through Java, based on which users looking for simulation of their algorithms should first review these classes and then make modifications depending on their requirements. Java programming language is a prerequisite to applying this simulator software. Moreover, CloudSim provides no graphical user interface, since it simply comprises a number of basic classes [25].

Hence, the newly proposed algorithm was simulated through CloudSim. In order to achieve better performance, the min-min algorithm was first simulated, followed by the new algorithm in two scenarios below.

1- The proposed algorithm with fault tolerance code (FLPMM)
2- The proposed algorithm without fault tolerance code (LPMM)

In large-scale workflows that may last for several months, the second scenario can help enhance the system performance since it increases fault tolerance. Due to parallel execution, this will not significantly escalate the execution time.

For better evaluation, three different scenarios, almost similar to the simulation details in [16], have been used in the execution of the algorithms. Tables 1,3,5 and Tables 2,4,6 represent the specifications of tasks and specifications of resources used in the scenarios.

Table 1. Tasks specification in senario A

Tasks	*Size(MIPS)*	*Cost($)*	*Lim(ms)*	*Priority*
T1	100	100	110	Ordinary
T2	150	10	500	Ordinary
T3	200	10	250	Ordinary
T4	250	20	260	VIP
T5	500	50	700	Ordinary

Table 2. Resources specification in senario A

Virtual machines	*Cpu(MIPS)*	*Ram(MB)*
VM0	100	256
VM1	80	256
VM2	50	256

In scenario A, number of resources is chosen to be 3 and number of tasks is chosen to be 5 with different proportion of VIP tasks.

Table 3. Tasks specification in senario B

Tasks	*Size(MIPS)*	*Cost($)*	*Lim(ms)*	*Priority*
T1	160	160	180	VIP
T2	170	170	200	VIP
T3	320	320	450	VIP
T4	40	40	100	VIP
T5	360	360	460	VIP
T6	40	40	50	VIP
T7	800	800	1100	VIP
T8	100	1000	600	VIP
T9	120	120	800	Ordinary
T10	630	630	900	Ordinary

Table 4. Resources specification in senario B

Virtual machines	*Cpu(MIPS)*	*Ram(MB)*
VM0	70	256
VM1	80	256
VM2	40	256
VM3	70	256
VM4	50	256

In scenario B and C, number of resources is chosen to be 5 and number of tasks is chosen to be 10 with different proportion of VIP tasks.
The tasks and resources specification for simulating Scenario C are listed in Table 5 and table 6.

Table 5. Tasks specification in

Tasks	*Size(MIPS)*	*Cost($)*	*Lim(ms)*	*Priority*
T1	950	950	1200	VIP
T2	240	240	1000	VIP
T3	610	610	1300	Ordinary
T4	490	490	800	Ordinary
T5	890	890	1800	Ordinary
T6	760	760	1500	Ordinary
T7	460	460	490	Ordinary
T8	30	30	60	Ordinary
T9	820	820	1500	Ordinary
T10	450	450	850	Ordinary

Table 6. Resources specification in senario C

Virtual machines	*Cpu(MIPS)*	*Ram(MB)*
VM0	10	256
VM1	80	256
VM2	90	256
VM3	50	256
VM4	30	256

The following figures 4-6 represent the performance of resources under their three respective algorithms.

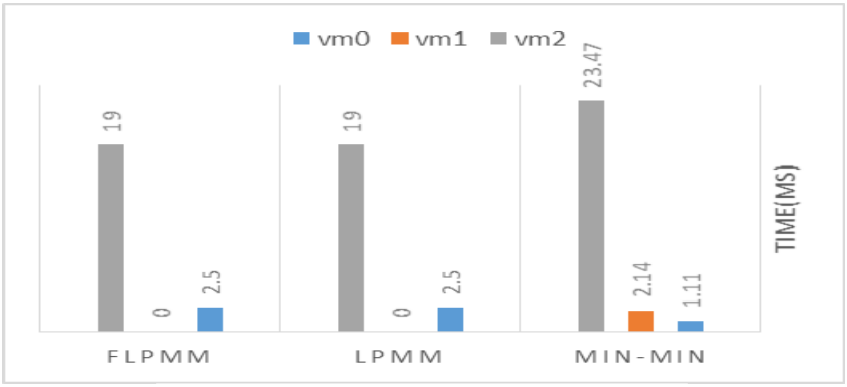

Fig 4. Performance of Resources in Senario A

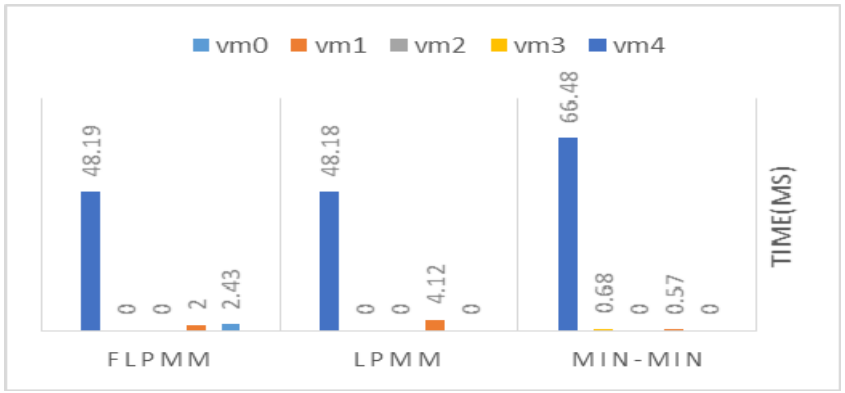

Fig 5. Performance of Resources in Senario B

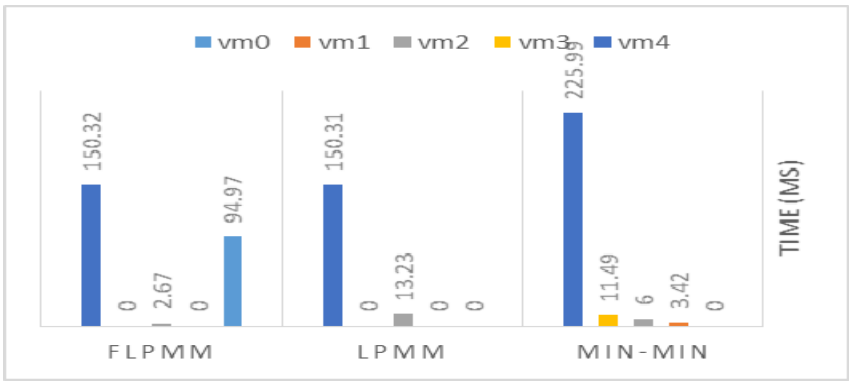

Fig 6. Performance of Resources in Senario C

Figure7 shows the execution time of the three scenarios in above algorithms.

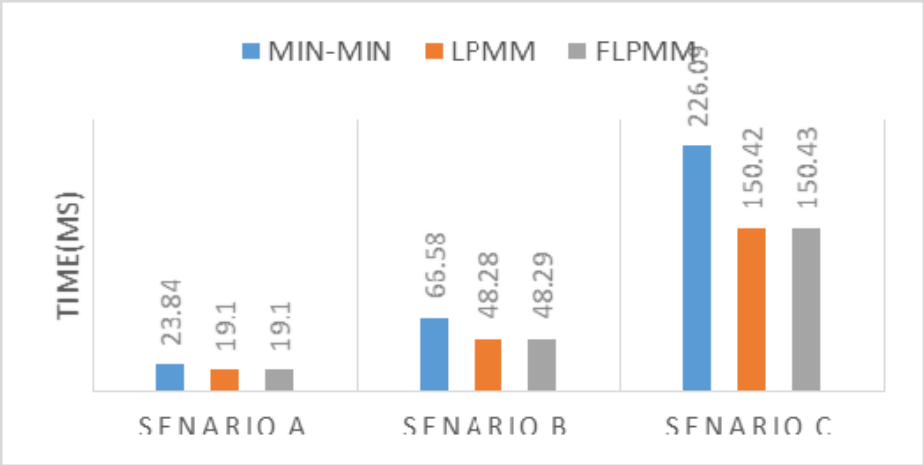

Fig 7. Makespan Results

According to the results obtained by simulation of algorithms, the newly proposed algorithm performed far better in the first and second scenarios with a slight difference in execution time, as compared to min-min algorithm. Given that fact that min-min algorithm does not entail any policies of fault tolerance, smart cost payment, prioritizing tasks and load balancing. It has longer execution time compared to the proposed algorithm which covers all those parameters. As a result, the new comprehensive algorithm can be considered ideal since it entails most of the key evaluation parameters suitable for a dynamic cloud environment.

5 CONCLUSIONS

Scheduling algorithms have a lot to do with the efficiency of cloud computing environments through selection of suitable resources and assignment of workflows to them. Given the factors affecting their efficiency, these algorithms try to use resources optimally and increase the efficiency of this environment. This study attempted to present a new algorithm based on min-min algorithm dubbed flpmm, which can respond to most requirements of users and cloud environments. Furthermore, it can be ideal for a dynamic cloud environment, enhancing system efficiency. The parameters considered in this algorithm can fulfill the requirements of both users and cloud environments, such as meeting the deadlines, cost constraints, prioritizing tasks, load balancing and enhancing fault toleranc. The noteworthy point is that the newly proposed algorithm managed to complete all workflows based on estimating most requirements within a period shorter than that of min-min algorithm which focuses only on reducing the deadline. By taking into account the key parameters of load balancing, the new algorithm can greatly contribute to system performance, managing to utilize the resources optimally and complete the workflows within the shortest possible time. Given the findings above and the results obtained by simulation of the newly proposed algorithm compared to its counterparts, it can be concluded that flpmm algorithm yielded far better results through fulfilling most requirements of users and cloud environments, only a few of which had been previously met by other algorithms.

REFERENCES

[1] Yonghong luo and shuren zhou , ” power consumption optimization strategy of cloud workflow scheduling based on sla”wseas transactions on systems, yonghong luo, shuren zhou, e-issn:2224-2678, volume 13, 2014.

[2] Sara qaisar and kausar fiaz khawaja ,”cloud computing: network/security threats and countermeasures” , interdisciplinary journal of contemporary research in business - january 2012 vol 3, no 9.

[3] Bahman rashidi and mohsen sharifi and talieh jafari , "a survey on interoperability in the cloud computing environments" published online july 2013 in mecs (http://www.mecs-press.org/) doi: 10.5815/ijmecs.2013.06.03.

[4] Pankaj sareen , " cloud computing: types, architecture, applications, concerns, virtualization and role of it governance in cloud" , international journal of advanced research in computer science and software engineering ,volume 3, issue 3, march 2013.

[5] Minoo soltanshahi and aliakbar niknafs, " a study on factors contributing to efficiency of scheduling algorithms in a cloud computing environment; overview of several algorithms", ciência e natura, v. 37 part 2, p. 427−433, december 2015, doi: http://dx.doi.org/105902/2179460x.

[6] Kennedy, j. And eberhart, r. C., "particle swarm optimization", proceedings of ieee international conference on neural networks, piscataway, nj, pp. 1942-1948, 1995.

[7] Tracy d, braun, "a comparison of eleven static heuristics for mapping a class ofindependent tasks onto heterogeneous distributed computing systems" journal of parallel and distributed computing , volume 61, issue 6, pages 810 – 837, 2001.

[8] Saeid abrishami and mahmoud naghibzadeh and dick h.j. Epema ," deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds " , future generation computer systems 29 (2013) 158–169

[9] Pooja, naveen kumari ," performance evaluation of cost-time based workflow scheduling algorithms in cloud computing " , international journal of advanced research in computer science and software engineering , volume 3, issue 9, september 2013 issn: 2277 128x.

[10] Hamo.a and saeed.a, towards a reference model for surveying a load balancing, ijcsns international journal of computer science and network security , vol.13, 2013.

[11] Jayadivya s k and jaya nirmala s and mary saira bhanu s ," fault tolerant workflow scheduling based on replication and resubmission of tasks in cloud computing "jayadivya s k et al. / international journal on computer science and engineering (ijcse), issn : 0975-3397 vol. 4 no. 06 june 2012 .

[12] Yang xu, lei wu, liying guo, zheng chen , lai yang and zhongzhi shi ," an intelligent load balancing algorithm towards efficient cloud computing " , ai for data center management and cloud computing: papers from the 2011 aaai workshop (ws-11-08).

[13] Zhong xu and rong huang, "performance study of load balanacing algorithms in distributed web server systems", cs213 parallel and distributed processing project report 2009.

[14] P.warstein and h.situ and z.huang, "load balancing in a cluster computer" in proceeding of the seventh international conference on parallel and distributed computing, applications and technologies ieee 2010.

[15] Parveen jain and daya gupta ," an algorithm for dynamic load balancing in distributed systems with multiple supporting nodes by exploiting the interrupt service" , international journal of recent trends in engineering, vol 1, no. 1, may 2009.

[16] Huankai chen, professor frank wang, dr na helian and gbola akanmu ," user-priority guided min-min scheduling algorithm for load balancing in cloud computing" , conference paper · february 2013 doi: 10.1109/parcomptech.2013.6621389.

[17] Er. Rajeev mangla , er. Harpreet singh , "recovery and user priority based load balancing in cloud computing" , international journal of engineering sciences & research technology , issn:

2277-9655 scientific journal impact factor: 3.449(isra), impact factor: 2.114 , mangla, 4(2): february, 2015.

[18] Sran.n and kaur.n " comparative analysis of existing load balancing techniques in cloud computing" , international journal of engineering science invention, vol.2, issue 1,2013.

[19] S. Raogururaj, s. Stoneharold and t.c. Hu, "assignment of tasks in a distributed processor system with limited memory", ieee trans. On computers, vol. C- 28, no. 4, april 1979.

[20] T. Kokilavani, dr. D.i. George amalarethinam, "load balanced min-min algorithm for static meta-task scheduling in grid computing", international journal of computer applications (0975 – 8887), volume 20– no.2, april 2011.

[21] Yalda aryan and arash ghorbannia delavar ," a bi-objective workflow application scheduling in cloud computing systems", international journal on integrating technology in education (ijite) vol.3, no.2, june 2014.

[22] O. Ibarra and c. Kim, "heuristic algorithms for scheduling independent tasks on non-identical processors", journal of the acm, 24(2):280{289, 1977. Issn 0004-5411.}

[23] Salman meraji and m. Reza salehnamadi , "a batch mode scheduling algorithm for grid computing" , j. Basic. Appl. Sci. Res., 3(4)173-181, 2013 © 2013, textroad publication , issn 2090-4304 ,journal of basic and applied scientific research ,www.textroad.com.

[24] Distinguishing cloud computing from utility computing, 2008/3,(http:/ / www. Ebizq. Net / blogs/ saasweek/ 2008/ 03/ distinguishing_cloud_computing/).

[25] 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.", Software: Practice and Experience, 41(1): 23–50,2011.