Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment

Article *in* Computers & Electrical Engineering · November 2017

DOI: 10.1016/j.compeleceng.2017.11.018

2 authors:

MOHIT KUMAR
National Institute of Technology Jalandhar
25 PUBLICATIONS **159** CITATIONS
SEE PROFILE

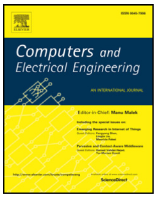
s. C. Sharma
Indian Institute of Technology Roorkee
115 PUBLICATIONS **856** CITATIONS
SEE PROFILE

Computers and Electrical Engineering 000 (2017) 1–17

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng



Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment[☆]

Mohit Kumar[*], S.C. Sharma

IIT Roorkee, India

ARTICLE INFO

Article history:
Received 22 April 2017
Revised 12 November 2017
Accepted 14 November 2017
Available online xxx

Keywords:
Cloud service provider
Task scheduler
Virtual machine
Load balancing
Makespan time
Elasticity

ABSTRACT

The most challenging problem for a cloud service provider is maintaining the quality of service parameters like reliability, elasticity, keeping the deadline and minimizing the makespan time as also the task rejection ratio. Therefore, the cloud service provider needs a dynamic task scheduling algorithm that reduces the makespan time while increasing the utilization ratio of cloud resources and meeting the user defined QoS parameters. In this paper, we have developed a dynamic scheduling algorithm that balances the workload among all the virtual machines with elastic resource provisioning and deprovisioning based on the last optimal k-interval. Further, the algorithm has been tested on variable number of tasks (10 to 30) to achieve better scalability. The computational results (Figs. 5–10) show that the developed algorithm decreases the makespan time and increases the task to meet the deadline ratio compared with the min-min, the first come-first-serve and the shortest-job-first algorithms in all conditions.

1. Introduction

Cloud computing provides the services either in the form of software application or hardware infrastructure on the basis of pay per use over the internet. It is the collection of heterogeneous resources that contain the characteristics of on demand self-service, scalability (scale-out and scale-up), resource pooling, broad network access, rapid elasticity and higher availability. It provides the different types of services like infrastructure as a service (IaaS), storage as a service (SaaS), software as a service (SaaS), platform as a service (PaaS) etc. These types of services are useful in scientific, business and industrial applications. User can send the request at any time for the resource to cloud service provider and cloud resource broker (cloud service provider) selects the best resource within user-defined deadline and budget. Cloud resource broker provides the on-demand service to the user. The number of users and applications are increasing gradually in cloud environment and in turn there is increase in the workload and traffic at the web applications which are deployed in the virtual machine (cloud resource). Therefore cloud resource broker needs an efficient algorithm that distributes the task fairly in all the running virtual machines and reduces the task rejection ratio so that the entire user task can be executed.

The main objective of the load balancing is to utilize the cloud resource in a manner that improves the average resource utilization ratio, response time and scalability of the web application. Efficient load balancing gives the minimum makespan time of tasks and increases the performance of the system. It also prevents bottleneck of the system which may occur due to load imbalance. Load balancing is one of the challenging research areas in the field of cloud computing whose

[☆] Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. L. Bittencourt.

[*] Corresponding author.

E-mail addresses: mohit05cs33@gmail.com (M. Kumar), scs60fpt@iitr.ac.in (S.C. Sharma).

https://doi.org/10.1016/j.compeleceng.2017.11.018
0045-7906/© 2017 Elsevier Ltd. All rights reserved.

objective is to balance the workload among the virtual machines. There are two steps of achieving the load balancing in cloud environment; first one is task scheduling and monitoring the virtual machine. Task scheduling is one of the best known optimization problems (NP Complete) in the field of computer science because cloud has heterogeneous resource (different host and virtual machine configuration) and very rapid on demand request change. Therefore it is difficult to predict and calculate all possible task-resource mapping in cloud environment. So we need an efficient task scheduling algorithm which can distribute the task in effective manner, so that less number of virtual machines faces overload or under-load condition. Second one is to monitor the virtual machine continuously and perform the load balancing operation (either task migration or virtual machine migration). Task migration approach has several advantages like over the virtual machine migration therefore we used the task migration approach in this paper. Cloud resource broker (CRB) monitor the virtual machine continuously in cloud environment. If any virtual machine is in overload or under-load condition after the task scheduling then cloud resource broker start the load balancing operation on the virtual machine and migrate the task from overloaded virtual machine to under loaded virtual machine.

In this paper we have analysed scalability (elasticity) aspect, which is the ability of a system to fit in a problem such that if scope of the problem increases (number of request increase, length of request vary randomly etc.). The ability of auto scaling on upcoming demands in cloud computing is biggest advantage for services provider as well as for user [1]. Auto scaling can reduce the risk, which is associated with request/load overflow causing server failure. Two types of auto scaling approaches are available in cloud environment i.e. reactive and proactive. Reactive approach gives their response to event after they have occurred. Reactive approaches are good for short term services but expensive for long term service. Proactive approach tries to eliminate the problem before they have chance to occur. A proactive scaling system that enables the services providers to schedule dynamic changes in the capacity that should be match with expected changes in application demand. To perform proactive scaling, firstly understand the expected change in workload i.e. one should try to understand that how much workload changes are possible from the excepted workload. Main drawback of proactive approach is that if predictive model is fails then resources will be in overutilization or underutilization mode and violates of the service level agreement (SLA). Proactive approach can be classified into three categories, cyclic, event, and prediction based. In this paper we are using prediction based approach that predicts the future demands on the basis of past history.

Scalability can be classified into two type's horizontal scalability (scale out) and vertical scalability (scale up). Vertical scalability can be achieved by making changes in the existing resources such as memory, hard drives, CPU's, etc. Vertical scalability is not generally used in cloud environment because common operating systems don't support these changes without rebooting on existing resources like memory, CPU's. Adding or releasing of one or more machine instance or computing node of same type is called horizontal scaling. Adding the IT resource in horizontally is called scale-out and releasing the IT resource horizontally scale-in. horizontal scaling is better than vertical scaling in cloud environment because it is less expensive and not limited by hardware capacity.

The reminder of this paper is organized as follows: Section 2 describes the related work in which we will discuss existing load balancing algorithm and technique in cloud environment that is related to present research work, Section 3 holds the problem formulation, Section 4 describes the proposed architecture section 5 presents the dynamic load balancing algorithm with elasticity, Section 6 shows the Analysis and comparison of results and Section 7 conclusion and future scope of the paper work.

2. Related work

Several static [2,5–6] and dynamic algorithms [7,12–19] for load balancing have been proposed in last decade. Static algorithm needs advanced information about the upcoming number of request (task) and information about available cloud running virtual machine (cloud resources). There is no need to monitor the cloud resources continuously because this type of algorithm works well when node has low variation in workload. It is difficult to predict the job execution time in cloud environment therefore job scheduler must be dynamic in nature. Existing job scheduling algorithms like conservative backfill, EASY etc. are unable to fill the resource gap efficiently. The work done by [2] focus at improving the backfill algorithm (IBA) not only improves the processing time of jobs but also provide the guarantee of quality of services in cloud environment. In this paper author's improve the IBA using balanced spiral (BS) method but this algorithm do not provide better processing time when job requests enter randomly in cloud environment. Dubey et al. [3] proposed an algorithm for metascheduler to solve the job scheduling problem in cloud computing and removed the limitation of IBA algorithm. In this paper, authors improved the processing time of upcoming jobs and resource utilization ratio of cloud resources considering priority of job as quality of service parameter. Sahoo et al. [4] proposed an algorithm based upon the greedy technique that reduces the makespan time and execution time of the tasks without using task migration or virtual machine migration approach for load balancing; proposed algorithm does not provide better results in real environment. First come first serve (FCFS) and shortest job first (SJF) [5,6] are static algorithms that are suitable for batch system. Static algorithm gives better results when there is low variation in workload. Literature review on dynamic based algorithm is reported in Table 1. Chen et al. [7] proposed a user guided min-min load balancing algorithm that not only minimize the execution time of the tasks but also remove the drawback of min-min algorithm (load is not properly balanced at each node). Proposed algorithm schedule n different length tasks to m heterogeneous cloud resource where scheduler chooses the minimum size task T_i and calculate the processing time at all the running virtual machine (resources) and assigns that virtual machine who can execute the task in minimum time. Task T_i is removed from the set S after the assignment and same procedure is repeated until all

Table 1
Literature review on dynamic based scheduling algorithm.

S N.	Year	Paper title	Parameter	Migration technique	Tool	Limitations
1	2015[3]	"A Priority Based Job Scheduling Algorithm Using IBA and EASY Algorithm for CloudMetaschedular"	Resource utilization, processing time	Apply heuristic approach to balance the load	Cloudsim	Task scheduling approaches don't give the guarantee of load balancing.
2	2016[12]	"Dynamic auto-scaling and scheduling of deadline constrained service workloads on IaaS clouds"	Execution time, makespan time	NA	OpenStack and cloudsim	Some important constraint is not considered like cost, heterogeneous virtual machine
3	2014[13]	"Brokering and Load-Balancing Mechanism in the Cloud À Revisited"	Response time, Processing time	NA	Cloud Analyst	Experiment is conducting on cloudsim, author is not ensuring that algorithm gives the same results/performance in real test bed environment.
4	2013[14]	"A Broker Based Architecture for Adaptive Load Balancing and Elastic Resource Provisioning and Deprovisioning in Multi-tenant Based Cloud Environments"	response time, throughput	Task migration	eucalyptus cloud	Limitation of this paper is load balancer unable to maintain the session in multi-tenant environment when same user request the multiple VM instances.
5	2014[15]	"Virtual machine selection and placement for dynamic consolidation in Cloud computing environment"	Energy consumption, reduce VM migration time	VM migration	Cloudsim	VM migration is costly and time consuming rather than task migration.
6	2016[16]	"Virtual machine resource scheduling algorithm for cloud computing based on auction mechanism"	profit of cloud service provider, resource utilization ratio.	NA	Cloudsim	When number of client is increase, large auction deadline interval will have a negative impact on the profit of the cloud service provider to some extent
7	2011[17]	"Deadline-constrained workflow scheduling in SaaS cloud"	computation time and cost with deadline	NA	Java based simulator	Proposed algorithm is implemented on Java based simulator that do not gives the guarantee of cloud environment, algorithm, specify the limitation of one variable do not gives the optimal multi objective solution.
8	2016[18]	"Deadline sensitive lease scheduling in cloud computing environment using AHP"	Resource utilization ratio, no. of leased scheduled, no of leased rejected	NA	Open nebula private cloud	Main limitation of backfilling algorithm is estimation of program execution must be known and it is static algorithm.
9	2015[19]	"Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds"	Ratio cost to budget, ratio make span to deadline	NA	Cloudsim	SPSS algorithm take lots of time for planning (10 min or more for 100 work flow)to distribute the work flow to virtual machine. While dynamic algorithm have high failure rate.
10		Proposed algorithm	Makespan time, task meet to deadline,	Task migration	Cloudsim	If avgofCount become large at the first iteration then it create more virtual machine than required.

upcoming tasks are assigned to the resources. Proposed algorithm is simulated at matlab toolbox to reduce the average task completion time and increase the average resource utilization ratio.

There are some dynamic algorithms which are using soft computing approach like Honey bee behaviour [8], particle swarm optimization (PSO) [9], ant colony optimization (ACO) [10], differential evaluation algorithm [11] etc. to solve the problem of load balancing. Honey bee behaviour inspired load balancing (HBB-LB) algorithm has been proposed by Babu et al. whose aim is to achieve the minimum response time and maximum throughput considering the priority of tasks in such a way that execution time of task should be minimum, drawback of this proposed algorithm is that if there are more priority based queue present then lower priority based load can wait for long time or might be possible to occur starvation

Table 2
Notations and description.

Notations	Description
$T_1, T_2, T_3. .. .. T_n$	Task request 1 to N submitted for schedule.
$R_1, R_2, R_3 \ldots\ldots R_n$	Cloud computing resource 1 to M are available for execution the task
MIPS	Millions of instructions per second
Sp	Represent the p^{th} schedule of workload (value of p is 1 to k)
Φ_{TiSp}	Represents the number of matched resources for task Ti for schedule Sp
$TT_{\mathrm{TiRj}\in\Phi\mathrm{TiSp}}$	Transfer time for task Ti on resource Rj in the matched cloud resource Φ_{TiSp}
$ET_{\mathrm{TiRj}\in\Phi\mathrm{TiSp}}$	Execution time of task Ti on matched resource Rj
$EET_{\mathrm{TiRj}\in\Phi\mathrm{TiSp}}$	Excepted execution time of resource Rj to execute the task Ti
TET(Sp)	Total execution time of task in schedule Sp
TL_{Ti}	Length of task Ti in Millions of instructions
p	Processing speed of resource R_j in MIPS
B_{Rj}	Bandwidth of resource Rj
V_{TiRj}	Decision variable is used to represent weather resource R_j is in matched resource list for the task Ti
D_{Ti}	Deadline of task Ti
FT_{Ti}	Finishing time of task Ti
WT_{Ti}	Waiting time of task Ti
TM_{Sp}	Number of task match with resource in schedule Sp
q	Number of cpu's
W_{Rj}	Workload available on resource R_j
MET_{Rj}	Maximum execution time of resource R_j
MST	makespan time

problem, another meta heuristic technique to solve the problem of load balancing in cloud computing is based on Task based system load balancing using particle swarm optimization technique (TBSLBPSO), Fahimeh et al. defines the objective function based on minimization the makespan time and task transfer time in which transferring the extra task from an overloaded virtual machine to under loaded (or ideal) virtual machine rather than migrate the virtual machine from one physical machine to other physical machine. PSO is population based stochastic optimization technique that searches the ideal or busy virtual machine in minimum time rather than conventional algorithm but these meta heuristic technique gives approximate results not exact results and these techniques do not work well in local optima i.e., these algorithm can get trapped in local optima. Pacini et al. [10] used ant colony optimization algorithm to optimize the response time and throughput parameter using the virtual machine migration approach but virtual machine migration technique is costly and time taking therefore ACO is not better than honey bee and particle swarm optimization but performs better than differential evolution algorithm, Tsai et al.[11] proposed the algorithm to reduce makespan time and cost using evolution algorithm(EA), EA is different optimization algorithm from classical optimization algorithm, Its main variation operator is mutation, it uses other operation also like randomness, crossover and selection to optimize the objective function but depends upon the problem (objective function). Main drawback of EA algorithm is that it never knows for certain when to stop.

Table 1 summarises the research papers related to the present work in terms of type of dynamic scheduling algorithm, tool used to implement the algorithm, performance metrics migration technique and limitation of the proposed algorithm.

It is observed from above the table that all the algorithms have used different approached, tools, parameter and have some pros and cons.

Specific contribution of this paper includes:

- We have developed a scheduling algorithm based on the last optimal k-interval that balances the workload among all the virtual machine with elastic resource provisioning and deprovisioning which overcome the drawback of algorithm proposed by Somasundaram et al. [14].
- The proposed algorithm distributes the task and adds the cloud resource if task rejection ratio is more than the SLA defined threshold value.
- The proposed algorithm monitors the load at each virtual machine and data centre continuously. If any virtual machine is overloaded then find the under loaded virtual machine and transfer the task from over loaded virtual machine to under loaded virtual machine using the task migration policy.

3. Problem formulation

For efficient load balancing schedule all the jobs (tasks) to cloud resource (virtual machine) in such a ways that cloud user can execute their task in minimum time (within the deadline) and resource utilization should be maximum i.e., cloud user is excepted to minimum makespan time and minimum cost while cloud service provider expectation is to utilize the resource maximally. Cloud resource broker received N number of task request T_1, $T_{2,}$ T_3 . . . . T_N, Which are independent in nature and non priority basis. Every task has task length TL_{Ti} which is expressed in MI (million instructions) and deadline of each task is D_{Ti} .

Every task requires p processing speed, q number of cpu, r amount of main memory and required Table 2 notations and its descriptions which are used in equations, bandwidth B in MBPS. Cloud resource broker have M number of resources (virtual machine) R_1, R_2, R_3.. . . . . R_M, which are heterogeneous in nature in terms of processing speed, memory, bandwidth etc. Matchmaker try to match each task Ti to virtual machine R_j(value of j is 0 to M-1), if a resource R_j is match with task Ti then value of decision variable V_{TiRj} is 1 otherwise its value is 0. Cloud resource broker contain the matched resource list (φ) of tasks available in a schedule (Si) that may contain all the cloud resources M or less than M.

The main aim of objective function is to minimize the makespan time of scheduling algorithm considering deadline as constraint. Our problem is based on single objective function therefore Objective function minimize the total execution time of tasks T_1, $T_{2,}$ T_3 . . . . T_N submitted in a particular schedule Sp as shown in Eq. (1).

Objective function:

$$\mathrm{Min}\, TET_{Ti} = EET_{\mathrm{TiRj}\in\Phi\mathrm{TiSp}} \tag{1}$$

$$ET_{\mathrm{TiRj}\in\Phi\mathrm{TiSp}} = TL_{Ti}/p * q \tag{2}$$

$$TT_{\mathrm{TiRj}\in\Phi\mathrm{TiSp=}}\ TL_{Ti}/B_{Rj} \tag{3}$$

$$EET_{\mathrm{TiRj}\in\Phi\mathrm{TiSp}} = ET_{TiRj} + TT_{TiRj} \tag{4}$$

Where $|\Phi\mathrm{TiSp}| \leq \mathrm{M}$

Excepted execution time is the sum of execution time and task transfer time. Number of match resource for task Ti may be less than or equal to total number of available resource in cloud environment. When user submits the task with deadline, its task execution time is depend that how much workload is available on that resource. The time required to complete the task on available cloud resource is expressed in Eq. (5) and task will be assigned to that resource that satisfied the condition shown in Eq. (6).

$$RT_{TiRj} = \ D_{Ti} - W_{Rj} \tag{5}$$

$$ET_{TiRj} \leq RT_{TiRj} \tag{6}$$

Here W_{Rj} represent the available workload on resource before assigned the Task Ti. Workload on virtual machine (cloud resource) at a particular time can be calculated by Eqs. (7) and (8) [8].

$$L_{V,Mi,T} = K * TL_{Ti(t)}/S_{(V,Mi,t)} \tag{7}$$

Where K = {1,2,3.................. N}

$S_{(V,Mi,\ t)}$ is defined the service rate of virtual machine at time t, that can be expressed in the form of processing power (p) and number of cpu (q) at a time t.

$$\text{Equation can be defined as}\, S_{(V,Mi,t)} = p * x_{(t)} \tag{8}$$

Where x = {1,2,3........ .....q}

Load on a virtual machine at a time t can be calculated as the number of task on particular virtual machine is divided by service rate of that virtual machine VM. Total load on all available virtual machine can be calculated as

$$L = \sum\nolimits_{j=1}^{M} L_{VMj} \tag{9}$$

After assigning the tasks to virtual machine workload on virtual machine will be increase. We can calculate the total workload by Eq. (10)

$$W_{Rj} = W_{Rj} + ET_{TiRj} \tag{10}$$

Cloud data centre contain M number of resource and each resource contain one or more than one task therefore we have to find the task completion time of all resource after that find the makespan time of resource. Makespan time is the largest time of virtual machine that is required to execute all the tasks/job.

$$\begin{matrix} MET_{Rj} = \sum EET_{TiRj} \\ Ti \in \varphi_{Rj} \end{matrix} \tag{11}$$

$$\text{Makespan time (MST)} = \max(MET_{Rj}) \tag{12}$$

Subject to:

$$FT_{Ti} > a_{T_i} + ET_{TiRj} \quad i \in N \tag{13}$$

FT_{Ti} is the finishing time of task Ti at virtual machine VM_j. a_{Ti} is task arrival time, if it is known and certain then problem is static otherwise problem is dynamic. ET_{TiRj} is the execution time of task Ti at virtual machine VM_j. Eq. (13) indicates that a task can't be started before its time.

$$FT_{i,R_j} \geq FT_{i-1,R_j} + ET_{\mathrm{TiRj}} \tag{14}$$

Eq. (14) represent that a task start to execute at a virtual machine only when previous task has been completed its execution at that particular virtual machine.

Find out the capacity of virtual machine (resource) to know that how many number of virtual machine are overloaded condition and under loaded condition. If any virtual machine is overloaded then transfer the task to under loaded virtual machine, if large number of virtual machine are in under loaded condition then check the condition and reduce the virtual machine.

$$\text{Capacity of virtual machine}\, C_{Rj} = p * q \tag{15}$$

After that find out the average number of task unable to meet deadline in all interval because on the basis of those task number of virtual machine can be increase or decrease if more number of task are unable to meet deadline then create new virtual machine. Firstly we find number of task unable to meet deadline in each interval with the help of count variable and add the value of count after every interval (iteration) with the help of arraylist. Initially declare the variable avgofCount and sumofCount and finally find the average number of task unable to meet deadline after each interval rather than completing the entire interval.

$$\begin{aligned}
&\mathit{listAvg}.\text{add}(\text{count});\\
&\text{declareavgofCount} = 1, \text{sumofCount} = 0;\\
&\textbf{for}(\text{Integerindex} : \mathit{listAvg})\\
&\{\text{sumofCount} = \text{sumofCount} + \text{index};\}\\
&\text{avgofCount} = \text{sumofCount}/\mathit{listAvg}.\text{size}();
\end{aligned} \tag{16}$$

4. Proposed architecture

The proposed architecture of cloud resource broker for dynamic task scheduling and load balancing with elasticity is represented in Fig 1 which is a modified architecture proposed by Somasundaram et.al. [14]. the brief description of the proposed architecture is given here.

4.1. Job request handler

Cloud user can access the cloud services (resources) from anywhere and anytime. Cloud contains huge number of datacenters that are distributed in geographical area. When a user submits the request for service, request goes to nearest datacenter. User submit the job J1,J2……..Jn through the graphical user interface or web interface with service requirement in terms of quality of service (QoS), hardware, software etc. cloud user requirement vary dynamically in terms of QoS (throughput, efficiency, user satisfaction, deadline, response time etc.), hardware (number of processor required, memory required, bandwidth required etc.) and software (Mpich-1.2.7, FFTW-3.X etc.). When user submit the job to the cloud, it is accepted by job request handler (gatekeeper), job request handler check the request at verify node to know who is sending the request, a legitimate user or an attacker, if request is coming from legitimate user then it is sent to the controller otherwise it discards the request. Cloud resource cannot be access directly in real environment but can be accessed through RESTFUL web API /SOAP, the main challenge is to allocate the resource to end user because user request is unpredictable and change at run time based on their application.

4.2. Controller node

This is the main component of cloud resource broker. job request handler forward the authentic request to controller node for further processing, controller node send the all application (job) request to matchmaker and also handles the incoming and outgoing request from other the components like job request handler (JRH), cloud resource provisioner (CRP), dynamic task scheduler and load balancer (DTSLB), cloud load and resource information (CLRI) aggregator.

4.3. Matchmaker

It contains the information about all the virtual machines (which virtual machine is in idle condition or which one is busy) and user job request. When matchmaker receives the request from the controller it checks the following quality of service parameter:

- Upcoming request is priority based or non priority based
- Upcoming request has a deadline or not

User required parameter (response time, makespan time, resource utilization etc.)

Matchmaker map the user request to available running virtual machine as per application requirement and forward the mapping list to task scheduler and load balancer component.

Please cite this article as: M. Kumar, S.C. Sharma, Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.018

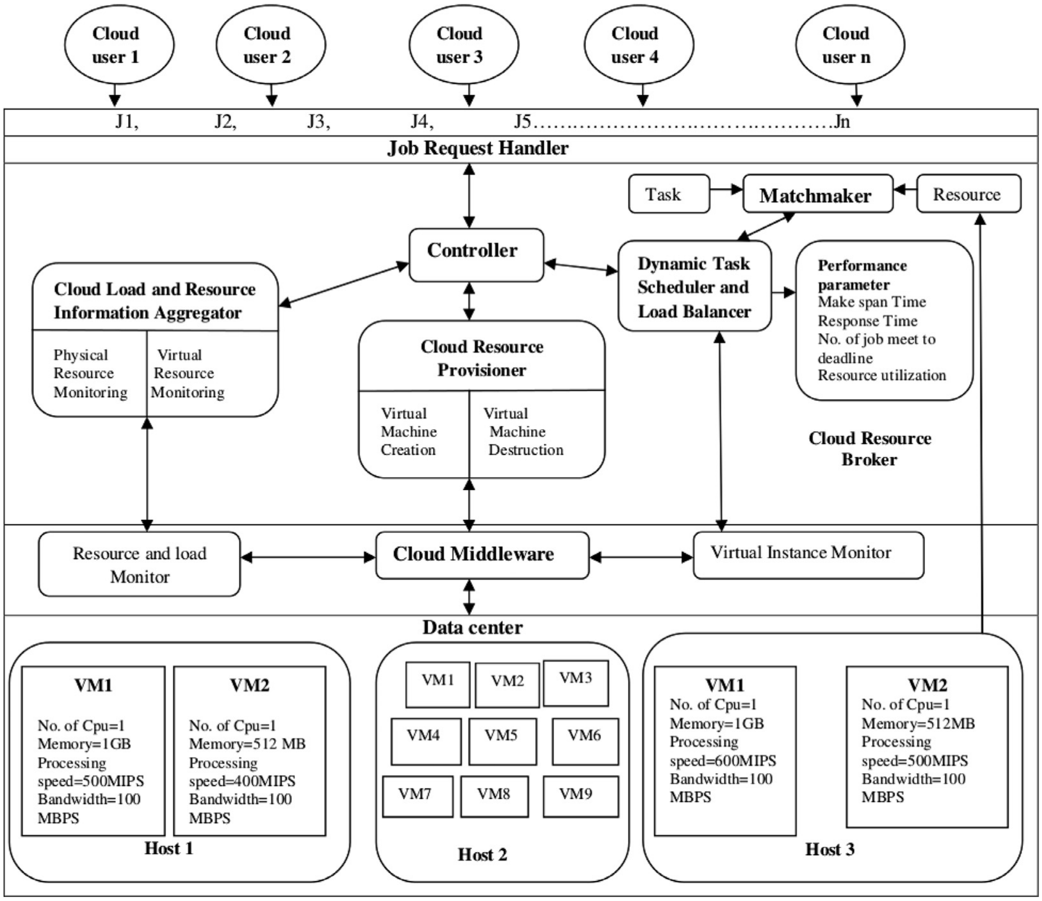


Fig. 1. Proposed cloud resource broker architecture.

4.4. Dynamic task scheduler and load balancer

Dynamic task scheduler receives the mapping list from the matchmaker and allocates the task (job) to the virtual machine as per scheduling algorithm. There are lot of task scheduling and load balancing algorithms present in cloud computing that works on parameters like makespan time, response time, throughput, resource utilization, cost etc. Task scheduler schedules the task in such a ways that all tasks are completed in minimum time. Haizea works as resource scheduler in private cloud OpenNebula. Cloud load balancer monitors all the virtual machine and compute the load on all the virtual machine. If any virtual machine is in overloaded condition then the task of that virtual machine is migrated to other under loaded virtual machine. If overloaded virtual machine does not exist then task is migrated to the balance virtual machine which can execute the task in minimum time. If load at the running virtual machine is more than the threshold value then it invoke the CRP and create new virtual machine. If load at the running virtual machine is below the threshold then it invoke the CRP for destroying the virtual machine.

4.5. Cloud load and resource information aggregator

CLRI is mainly used for aggregating the resource information like memory, processor, bandwidth, load etc. from CSP's. It continuously monitors the resource and collects the information about the resource. It interacts with Cloud Monitoring and Discovery Service (CMDS) to retrieve the information about cloud resource. CMDS is used to monitor the virtual machine (idle or busy state) and discover the resource information. CLRI trigger a query (request) about the virtual machine, CMDS collect the information from available virtual machine and reply to CLRI.

4.6. Resource and load monitor

It is mainly used to monitor the private cloud (such as open Nebula, eucalyptus etc.) resource. Ganglia work as external information provider about the host and virtual machine in open Nebula. It collects the information about the resources and passes to the resource and load monitor component.

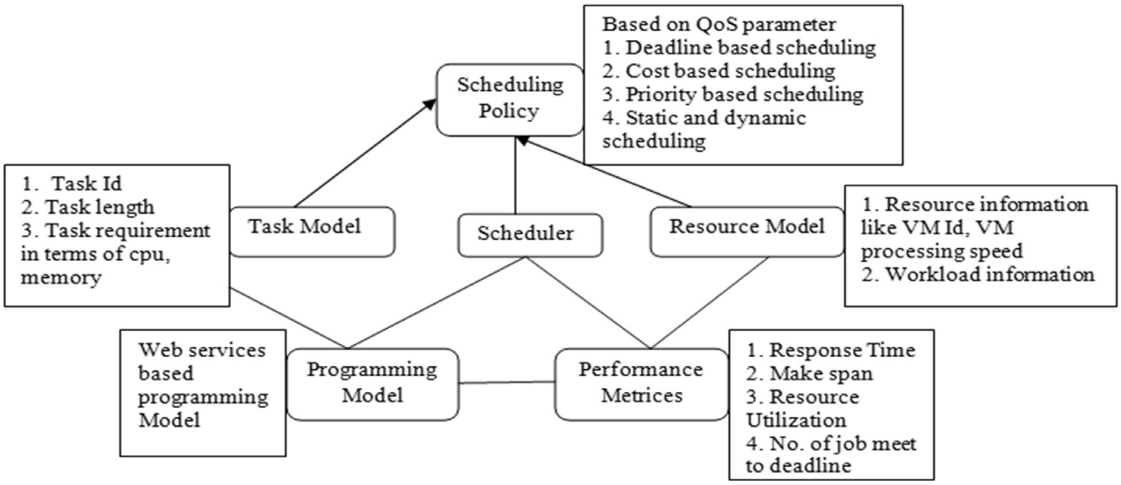

Based on QoS parameter
1. Deadline based scheduling
2. Cost based scheduling
3. Priority based scheduling
4. Static and dynamic scheduling
Scheduling Policy
1. Task Id
2. Task length
3. Task requirement in terms of cpu, memory
Task Model
Scheduler
Resource Model
1. Resource information like VM Id, VM processing speed
2. Workload information
Web services based programming Model
Programming Model
Performance Metrices
1. Response Time
2. Make span
3. Resource Utilization
4. No. of job meet to deadline

Fig. 2. Cloud task scheduling model.

Table 3
VM Properties.

VM Id	VM MIPS	VM Image size	Memory	No. Of cpu	Hypervisor
0	500	1000	256	1	Xen
1	520	1000	512	1	Xen
2	540	1000	512	1	Xen
3	560	1000	256	1	Xen
4	580	1000	256	1	Xen
5	600	1000	512	1	Xen
6	620	1000	256	1	Xen
7	640	1000	512	1	Xen
8	660	1000	215	1	Xen
9	680	1000	512	1	Xen

4.7. Cloud resource provisioner

The main objective of CRP is creation and deletion of virtual machine as per application (job) requirement. It interacts with cloud middleware (OpenNebula use extended version of D-Grid Resource centre Ruhr as middleware) for provisioning/deprovisioning of the virtual machine.

4.8. Virtual instance monitor

Monitors the load of each virtual machine continuously and forward it to load balancer that further passes it to controller node. There are two method of monitoring the resources

- *Event based*: When task is removed from a virtual machine or assigned to the virtual machine, after that monitor the status of all virtual machines. We are using event base approach.
- *Time based*: continuously monitor the resource after a particular time interval.

5. Dynamic load balancing algorithm with elasticity

Dynamic task scheduling algorithm is developed and discussed in this paper. The developed algorithm not only minimizes the makespan time but also increase the ratio of task meet to the deadline using elasticity concept. Cloud task scheduling model is shown in Fig. 2. We model our scheduling algorithm and analyse the performance of makespan time and number of task meet to deadline as per the parameters given in Tables 3 and 4. To develop this algorithm, we have created N number of task and length of task is generated randomly (range of task between 20000MI and 400000 MI) and created M number of heterogeneous virtual machine, each virtual machines have different processing power in terms of processor speed in MIPS, RAM etc.

Algorithm 1 sorts the task based on deadline. To test the algorithm we have taken two array, one have task length other have deadline of task; use the sorting algorithm to sort the task. The task which has minimum deadline value that task will be executed first because our aim is to execute more number of tasks before deadline expires. We proposed an efficient task scheduling algorithm that not only allocates the task to virtual machine (cloud resource) but also decrease the makespan

Please cite this article as: M. Kumar, S.C. Sharma, Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.018

Table 4
Task properties.

Task Id	Length	File Size	Output Size	No. of cpu required
0	130,795	300	300	1
1	224,339	300	300	1
2	48,212	300	300	1
3	330,838	300	300	1
4	269,322	300	300	1
5	65,245	300	300	1
6	383,678	300	300	1
7	263,607	300	300	1
8	137,286	300	300	1
9	328,394	300	300	1

Algorithm 1 for task sorting based on deadline.

```
1.  deadline[i] array represent the deadline of ith task
2.  Task [i] array represent the ith task
3.  Sort the task based on deadline
4.  For i = 1 to deadline_array length
5.  For j = i + 1 to deadline_array length
        if (deadline[j] < deadline[i])
        if condition is true
6.  Swap the content of deadline[j] with deadline[i]
7.  Swap the content (Task length) of Task[j] with Task[i]
```

Algorithm 2 for task scheduling based on deadline.

```
# EPT[VmSize] and MET[VmSize], min and a = 0 are variable
1.  Generate M number of virtual machine.
2.  We have to schedule N number of Task based on deadline.
3.  ∀Ti∈{$T_{1,}$ $T_2$, $T_3$.......$T_N$}
4.  At the starting number of task assigned to resource is null
        $\varphi_{RJ} \leftarrow$ {null}
5.  ∀$R_j$∈{$R_1$,$R_2$,$R_3$..........$R_M$}
6.  Find the execution time of task Ti at $R_j$
        $ET_{\mathrm{TiRj\in\Phi TiSp}}$ = EPT[j] = $TL_{Ti}$/p*q End for loop of resource $R_j$
7.  Assigned the task $T_i$ to resource $R_j$ for minimum execution time
     ∀$R_j$ from 0 to M-1                EPT[j] = EPT[j] + MET[j];
     End of resource $R_j$ loop
     min = EPT[0];
     ∀$R_j$ from 0 to M-1
     if(min > EPT[j])                    a = j; min = EPT[j]
     if(deadline[Ti] > min) assigned the task to virtual machine with index a
     vm = getVmsCreatedList().get(a)
     MET[$a$] = EPT[$a$];
     End of for loop of resource $R_j$; End of for loop of tasks $T_{i,}$
8.  Load increase at resource $R_j$
     $W_{Rj} = W_{Rj} + ET_{TiRj}$
9.      $\varphi_{Rj} = \varphi_{Rj} \cup$ {Ti}
```

time of task as shown in Algorithm 2. Matchmaker and dynamic task scheduler component participate to allocate the task in effective way to virtual machine as shown in Fig. 1. Matchmaker match each task to available running virtual machine and pass the list of match resource to task scheduler that will allocate the task based upon scheduling algorithm. At the starting list of task assigned to resource is null shown in step 4 i.e. no task is assigned to virtual machine. To assign the task to virtual machine, we calculate the execution time of task at all the running virtual machine and current load at all virtual machine, compare it with deadline. If deadline of task is more than the execution time of task to virtual machine then assign the task to that virtual machine who can execute this task in minimum time. Load at the virtual machine is increased after assigning the task. It can be calculated by sum of previous available load and load of current assigned task. This process is continued until all tasks has not finished.

When dynamic task scheduler try to allocate the task to virtual machine, there are some tasks that do not fulfil the condition of deadline i.e., some tasks have execution time more than their deadline value therefore these type of tasks are unable to meet their deadline. These types of tasks are discarded in cloud environment. If large number of task is discarded then cloud service performance will degrade and user satisfaction level will also decrease. Therefore we proposed an algorithm that balances the load at all virtual machine and increase the ratio of task to meet with deadline using the elasticity concept (provisioning and deprovisioning of resource at run time) as shown in Algorithm 3. Flow chart of Algorithm 3 (Fig. 3) rep-

Algorithm 3 for Load Balancing decision with scalability.

```
UVM = under loaded VM, BVM = balanced VM, OVM = overloaded VM
count represent number of task unable to meet deadline
avgofCount represent avg. number of task unable to meet deadline, NumberofTask = N
1.  ∀Ti€{$T_{1,}$ $T_2$, $T_3$.......$T_N$}
2.  ∀$R_j$€{$R_1$,$R_2$,$R_3$..........$R_M$}
// Allocate the task $T_i$ to that resource $R_j$ who can execute in minimum time and find
out the min value as shown in step 7 in Algorithm 2
3.  If(deadline[$Ti$] > min)
      Assigned the task to virtual machine with index a
      Else count + + ;
4. Find avgofCount after each interval
   If(avgofCount > N * 0.25)
   Increase $R_j$ by 20%.
   Else If (avgofCount > N * 0.1)
   Increase $R_j$ by 10%.
   Else (avgofCount ≤ N * 0.1)
   No need to increase the virtual machine at current time
   End of for loop; End of for loop
5.  ∀$R_j$ ∀$T_i$, Calculate the load at each $R_j$
6.  Calculate the capacity($C_{Rj}$) of each $R_j$
7.  Find the number of OVM,UVM, and BVM
      OVM ≥ .9 * $C_{Rj}$, UVM < 0.2* $C_{Rj}$
8.  Sort the virtual machine in OVM decreasing order
9.  Sort the virtual machine in UVM in ascending order
10. Assigned the task of OVM to UVM and calculate the task transfer time.
11. Calculate the average number of UVM
      If(avgUVM > 0.25*$R_j$)
      Then reduce $R_j$ by 20%.
      Else If (avgUVM > 0.1*$R_j$)
      reduce $R_j$ by 10%
      Else (avgUVM ≤ 0.1*$R_j$)
      No need to decrease the virtual machine at current time
12.  End of for loop; End of loop
```

resents the threshold condition and virtual machine overloaded or underloaded condition briefly in well specified way. We calculated the number of task unable to meet deadline in each interval after that find the average of number of task that are unable to meet deadline in last k interval using the Eq. (16) and applied the condition that if more than 25% average number of task are rejected then increase the new virtual machine by 20%. If average rejected task is more than 10% then increase the virtual machine 10%. If average rejected task is less than or equal to10% then there is no need to increase the virtual machine at current time because sometimes instantaneous small peak load (less than 10%) can come at the virtual machine. We start to increase the virtual machine (up to 5% or 10%) which leads to unnecessary overhead because after sometime either I have to reduce the virtual machine or some virtual machine will be in under loaded condition.

We calculate the parameter avgofCount that is based upon last k interval and increase the virtual machine based upon the defined threshold value (25% or 10%) for next upcoming interval. Threshold value is defined at the time of service level agreement (SLA) based upon the parameter like upcoming number of request per minutes, workload, number of task unable to meet deadline etc. SLA is a contract between the service provider and users in which the Quality of Service (QoS) is defined. We have studied threshold based papers [14,20–24] related to our work for cloud environment and set the value of threshold based on the historical (Past) data. For this we have carried out various experiments with different value of threshold to find out the optimum results. Finally we have selected the threshold value that gives the optimum result. We find the overloaded, under loaded and balanced virtual machine after assigning the task to virtual machine. If any virtual machine is overloaded or under loaded condition then Sort the OVM and UVM in decreasing and increasing order and assign the task. The algorithm checks the condition of under loaded virtual machine if average of under loaded virtual machine is greater than the 25% of all available virtual machine then decrease the virtual machine by 20% for next interval. If it is more than by 10% then decrease the virtual machine 10% for next interval.

6. Analysis and comparison of results

The developed load balancing algorithm minimizes the makespan time and increase the ratio of task to meet deadline using the cloudsim platform. We choose cloudsim simulator [25] for experimental purpose because Cloudsim simulator is basically used for task scheduling and load balancing To test the algorithm we have created a datacentre that contains many number of host, and each host contains the number of virtual machines depending upon the configuration of host (processing speed, number of cpu, memory etc.) as shown in Fig. 4. Each virtual machine (VM) has their parameter like Id, MIPS, number of CPU etc. as shown in Table 3. After that we have generated cloudlet (Task) with their parameter like TaskId, length, filesize etc. as shown in Table 4. Cloud resource broker will submit bounded task to specific virtual machine (depends

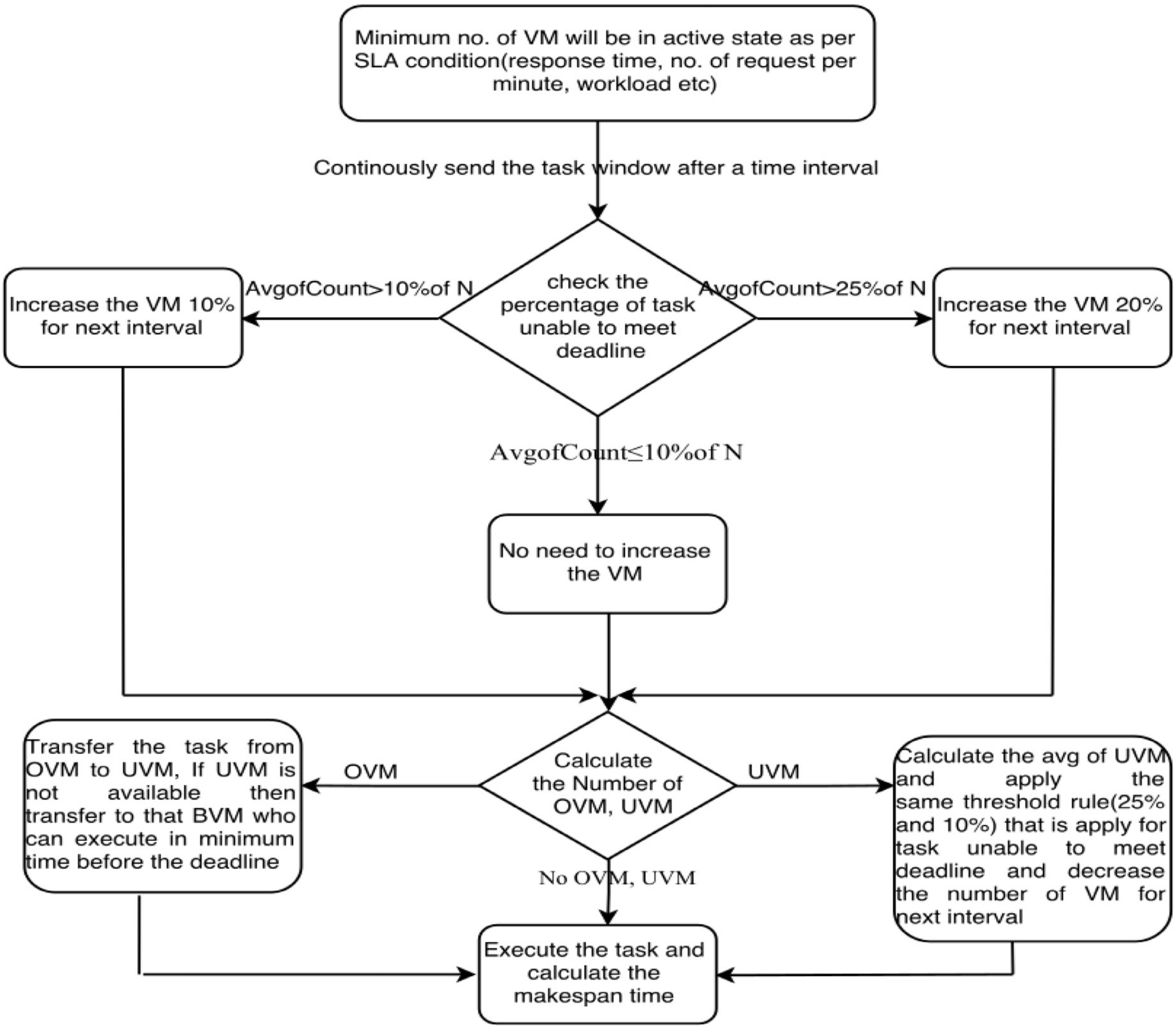

Minimum no. of VM will be in active state as per SLA condition(response time, no. of request per minute, workload etc)
Continously send the task window after a time interval
check the percentage of task unable to meet deadline
AvgofCount>10%of N
Increase the VM 10% for next interval
AvgofCount>25%of N
Increase the VM 20% for next interval
AvgofCount≤10%of N
No need to increase the VM
Calculate the Number of OVM, UVM
OVM
Transfer the task from OVM to UVM, If UVM is not available then transfer to that BVM who can execute in minimum time before the deadline
UVM
Calculate the avg of UVM and apply the same threshold rule(25% and 10%) that is apply for task unable to meet deadline and decrease the number of VM for next interval
No OVM, UVM
Execute the task and calculate the makespan time

Fig. 3. Flow chart of algorithm.

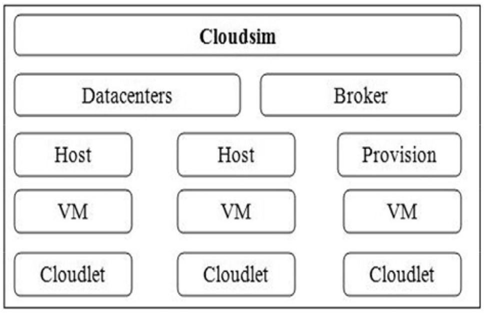
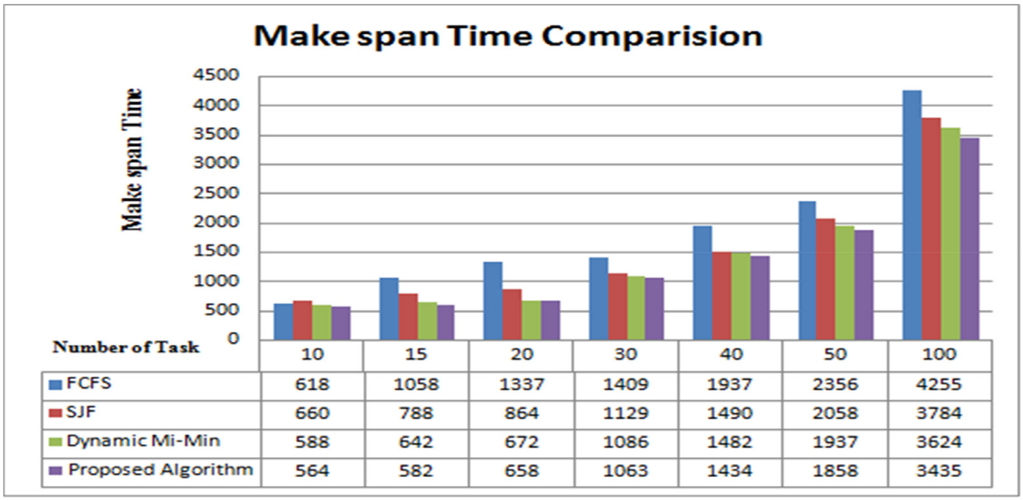


Fig. 4. Basic architecture of cloudsim.

upon the policy) with the help of broker. bindCloudletToVm(cloudlet.getCloudlet Id(),vm.getId()) method. In this paper, we have calculated and analysed makespan time and task unable to meet deadline with the help of cloudsim simulator.

6.1. Makespan time calculations

Let's consider the example; consider 10 virtual machine of different processing power and bandwidth of each virtual machine is 1000 MBPS, number of cpu for each virtual machine is one as shown in Table 3. The range of task is 10 to 100 and length of task is varying from 20000MI to 400000 MI as shown in Table 4. The numerical calculation to determine the



	10	15	20	30	40	50	100
🟦 FCFS	618	1058	1337	1409	1937	2356	4255
🟥 SJF	660	788	864	1129	1490	2058	3784
🟩 Dynamic Mi-Min	588	642	672	1086	1482	1937	3624
🟪 Proposed Algorithm	564	582	658	1063	1434	1858	3435

Fig. 5. Makespan time comparison between proposed algorithm vs FCFS, SJF, dynamic min-min.

Table 5
Task with deadline.

Task Id	Task length	Deadline of task	VM MIPS
0	319,775	500	500
1	43,753	720	520
2	50,077	780	540
3	276,496	685	560
4	133,858	745	580
5	45,215	620	600
6	132,290	490	620
7	367,951	450	640
8	291,873	700	660
9	385,050	660	680
10	93,020	870	
11	42,947	850	
12	30,362	590	
13	348,858	400	
14	69,942	560	

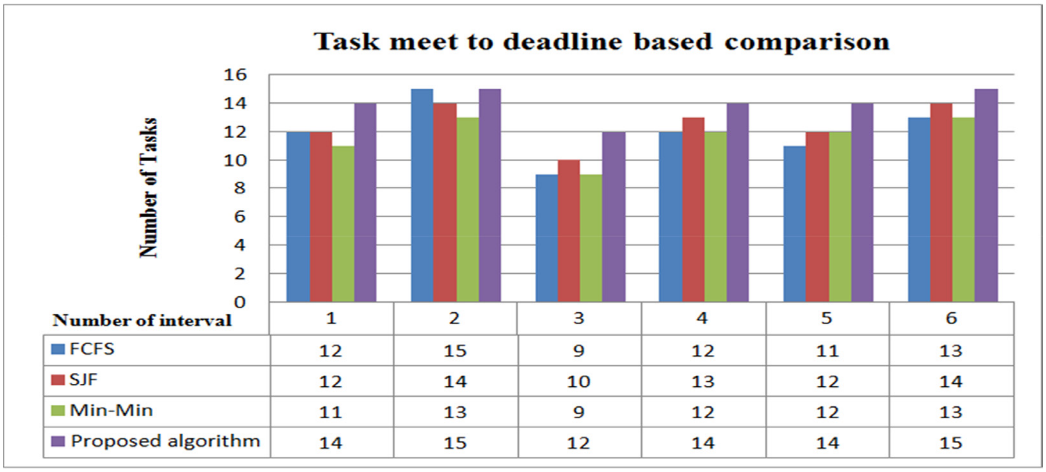
performance of the algorithm based upon the selecting the application (tasks) and resources instances and if application is memory-intensive that needed high-memory VM instances for database operation tasks. Therefore the proposed algorithm selects the length of task in a range (20000MI to 400000MI) and creates the virtual machine instance such that they can process the task. If the range of task is increased or decreased then results (makespan time, number of task unable to meet deadline and elasticity) are affected i.e. because same virtual machine instance either process the task early (underloaded condition) or delay (overloaded condition). Simulation has been run for more than 200 times on different number of task with random length and results are found using the space shared policy [25] in cloudsim.

Results of Tables 3 and 4 are graphically represented in Fig. 5, where x-axis represent the number of task and y-axis represent the makespan time of task in second. The developed scheduling algorithm allocates the task to all the virtual machine. The comparisons of makespan time with other popular scheduling algorithms like FCFS, SJF and dynamic min-min algorithm is shown in Fig. 5. Computational results shows that proposed algorithm reduces the makespan time of task compared to existing algorithm (FCFS, SJF and dynamic min-min algorithm) as shown in Fig. 5.

6.2. Number of task meets to deadline

To calculate the number of task meet to deadline we have created a window of 15 tasks with random length which is sent continuously to cloud resource broker after every 5 s interval. Different configuration of 10 virtual is created to process the upcoming task and deadline of task is created randomly (example shown in Table 5).

We computed the performance metric for analysing the number of task completed on or before the deadline specified by the user. Our proposed task scheduling algorithm considers the deadline as an important factor and find out the best resource for the task so that task can be executed before the deadline expired. Simulation has been run at different task with corresponding deadline and calculated results shows that proposed algorithm completes more tasks before deadline



Number of interval	1	2	3	4	5	6
🟦 FCFS	12	15	9	12	11	13
🟥 SJF	12	14	10	13	12	14
🟩 Min-Min	11	13	9	12	12	13
🟪 Proposed algorithm	14	15	12	14	14	15

Fig. 6. Number of task meet to deadline comparison between proposed algorithm vs FCFS,SJF, dynamic min-min.

Table 6
Task with deadline.

Task Id	Task length	Deadline of task	VM MIPS
0	381,771	785	500
1	392,397	745	520
2	339,760	920	540
3	323,461	690	560
4	272,332	750	580
5	325,970	700	600
6	333,911	660	620
7	182,561	2000	640
8	396,156	1200	660
9	215,744	1300	680
10	253,197	1250	
11	334,013	450	
12	344,630	400	
13	222,784	550	
14	253,745	1000	
15	153,285	800	
16	205,633	600	
17	98,049	1050	
18	107,218	300	
19	147,281	1150	

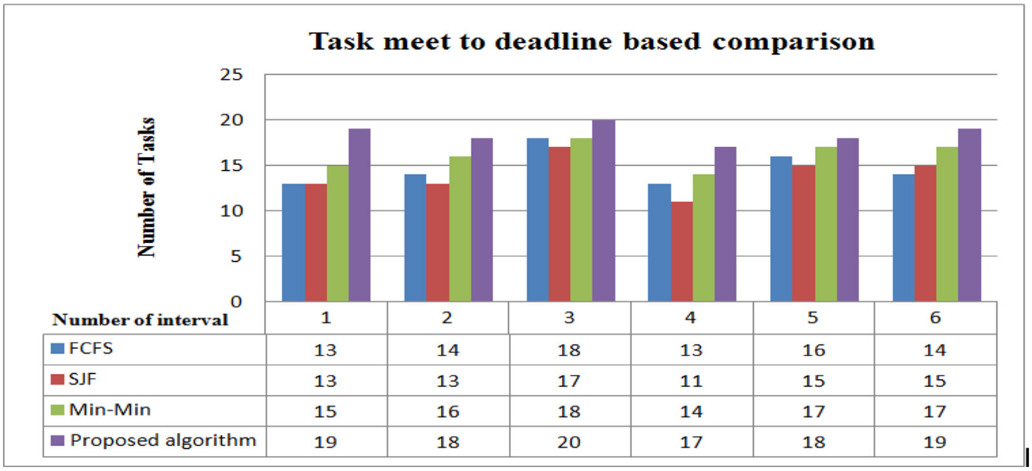
compare to FCFS, SJF and dynamic min-min as shown in Fig. 6. The proposed algorithm shows approximate 90% of the task meeting with deadline compare to 78% of FCFS, 81% of SJF and 76% of dynamic min-min algorithm.

Further we have tested our algorithm increase the number of task (15 to 20) with random length and deadline of task as shown in Table 6. Consider all the virtual machine has different processing capacity. When we run the simulation cloud resource broker send 20 tasks at every 5 s interval. The calculated results are shown in Fig. 7 which indicates that the proposed algorithm shows better results compare to in terms of number of tasks meet to deadline rather than FCFS, SJF and dynamic min-min algorithm.

6.3. Provisioning and deprovisioning (elasticity) of resource

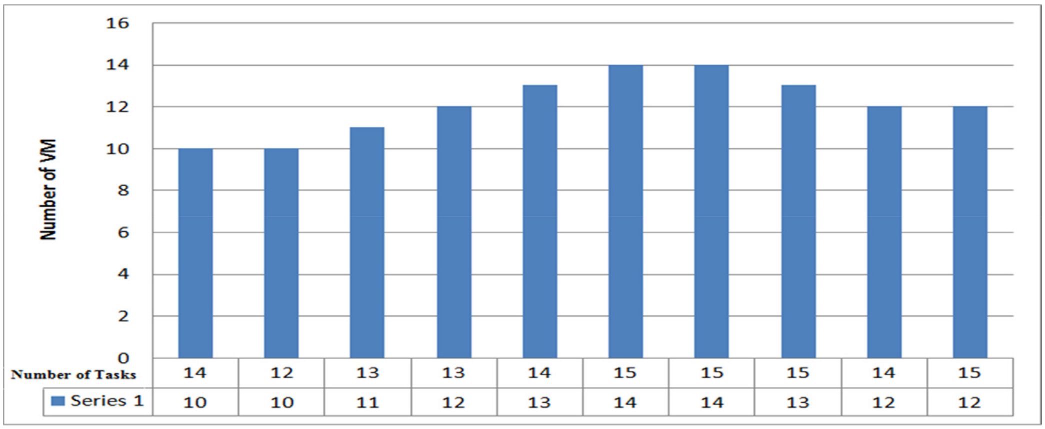
Elasticity is one of the important factors in cloud environment where user use the resources on the basis of pay per use and don't want to pay for resource which is not used by user. Elasticity is the ability to fit the resource needed to cope with dynamically upcoming load at the virtual machine. If load is increased at the resources then controller pass the instruction to cloud resource provisioner to increase resource in scale out fashion. When demand wanes, resource provisioner start to shrink back and remove unneeded resources. Let's consider the example in which we send continuously15 task of random length after 5 s interval (time interval will be large in real environment) and 10 virtual machine are ready to process the task at the initial phase. Deadline of task is given randomly. The proposed algorithm is tested with 15 number of task and the results show that 14 number of task meet deadline in first iteration, only one task is rejected (average is1) i.e. less than 10% of the total task.

Please cite this article as: M. Kumar, S.C. Sharma, Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.018



Number of interval	1	2	3	4	5	6
🟦 FCFS	13	14	18	13	16	14
🟥 SJF	13	13	17	11	15	15
🟩 Min-Min	15	16	18	14	17	17
🟪 Proposed algorithm	19	18	20	17	18	19

Fig. 7. Number of task meet to deadline comparison between proposed algorithm vs FCFS, SJF, dynamic min-min.



Number of Tasks	14	12	13	13	14	15	15	15	14	15
🟦 Series 1	10	10	11	12	13	14	14	13	12	12

Fig. 8. Provisioning and deprovisioning of cloud resource based on upcoming tasks.

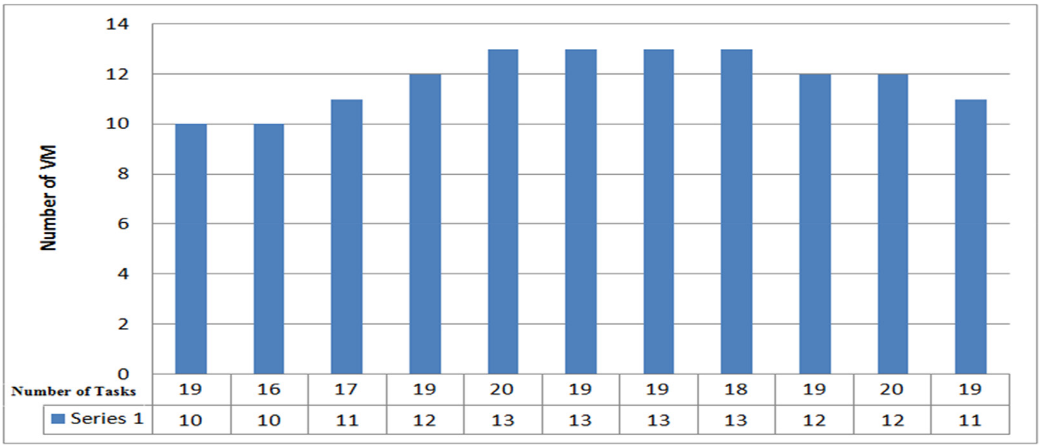
It shows that there is no need to increase the virtual machine. In next iteration three tasks have been rejected because of length of task is randomly selected (average become two) i.e. more than 10% task is rejected so cloud resource provisioner add 10% more virtual machine to process the upcoming tasks as per reported Algorithm 3. In the next iteration two tasks are rejected and calculated average task rejection is more than 10% so one more virtual machine is added. By randomly generated task, virtual machine is increases from 10to14 based upon the average number of task rejection but after some interval if workload start to decrease and some of the virtual machine goes to underloaded condition. The number of virtual machine is started to decrease as per underloaded threshold condition given in algorithm. After 10 intervals virtual machine is reduces from14 to 12 for executing the 15 random length tasks as shown in Fig. 8.

The algorithm is further tested with sending randomly window of 20 tasks. At the starting 10 virtual machine are ready to process the tasks. Simulation has been run for 11 intervals and it is observed that when the workload at the virtual machine is increases then the cloud resource provisioner create more virtual machines depending upon the condition given in Algorithm 3. If workload is decreased and virtual machine comes in under loaded condition then cloud resource provisioner decreases the virtual machine instances. Fig. 9 shows that virtual machine is increased from 10 to 13 at the time of high workload and decreased from 13 to11 at the time of low work load after the end of 11th iteration. Most of the existing approaches reported in the literature [20,23,24] take only single threshold limit for elasticity but we have considered 3 threshold limits (>25%, >10, <=10%) that gives better results than previous approaches. We have considered the average task unable to meet deadline in last k interval, while previous approach decide the elasticity only the basis of last interval. In the present paper, we have set the values for the threshold statically, however some authors set the threshold value dynamically [20,23,24].

6.4. Scalability

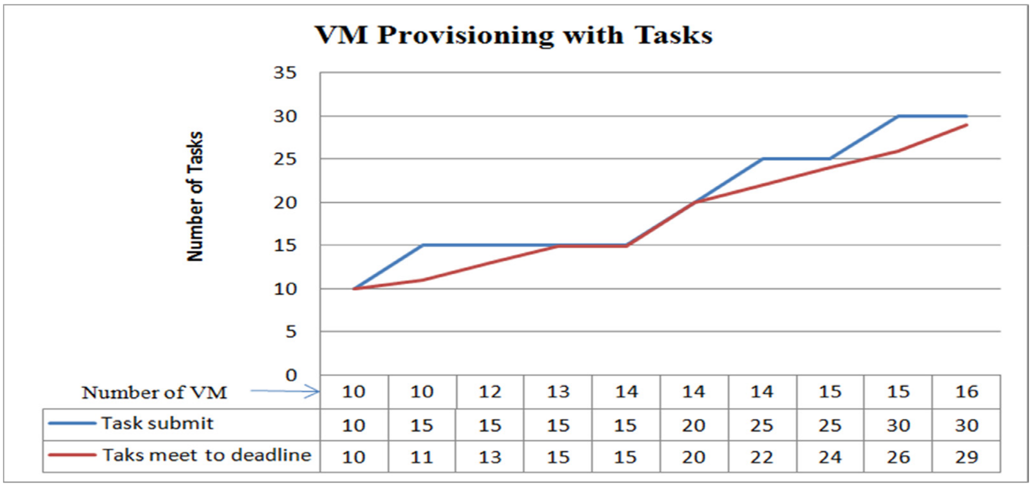
Scalability is the ability of the system to accommodate more loads by adding resource either horizontally (scale-out) or vertically (scale-up). Vertical scalability means that we are adding the additional hardware. This type of approach is apply

Please cite this article as: M. Kumar, S.C. Sharma, Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.018



Number of Tasks	19	16	17	19	20	19	19	18	19	20	19
■ Series 1	10	10	11	12	13	13	13	13	12	12	11

Fig. 9. Provisioning and deprovisioning of cloud resource based on upcoming tasks.



Number of VM	10	10	12	13	14	14	14	15	15	16
Task submit	10	15	15	15	15	20	25	25	30	30
Taks meet to deadline	10	11	13	15	15	20	22	24	26	29

Fig. 10. Scale-out of cloud resource based on upcoming task.

in web server, database servers etc. but limitation of vertical scaling is that hardware should be specific and how much memory, processor and disk a single server support. Therefore horizontal scaling is better than vertical scaling that add the number of node in horizontal scaling. We have tested our algorithm for horizontal scalability with number of tasks (10 to 30) with fixed length (200000 MI) and 10 virtual machines is process to execute the task at the starting of the simulation. On the basis of task rejection cloud resource provisioner increase the virtual machine instance and cloud environment provides the better scalability. We run the simulation (approximately 10 times) for 10 tasks at the starting and it are observed that all the tasks are easily executed by 10 virtual machines. If all the tasks executed by 10 virtual machine in first iteration then results are same for next all the iteration because all tasks have the same length. Now simulation has been run for 15 tasks and it is observed that only 11 tasks meet to deadline i.e. approximate 26% tasks has been rejected. So, cloud resource provisioner added the 20% virtual machine and it become 10 to 12 virtual machine. For the next iteration 20% average task has been rejected and virtual machine increase 12 to 13. This process is continuing until all the tasks have not been accepted as shown in Fig. 10.

Further the algorithm has been tested by increasing the tasks from 15 to 20, 25, 30 and same procedure is repeated. It is observed found that approximate 16 virtual machine is needed to execute the 30 tasks of same length. Fig. 10 represent that how many number of task we have submitted and how many number of task meet to deadline. y axis represent the number of task, x axis represent the number of virtual machine required to execute those task by using the proposed Algorithm 3. The overhead of the proposed algorithm is calculated based on the value of k which should not be more than last15 intervals. We have analysed the performance of the algorithm till last 30 intervals, as after 15 intervals its performance start to degrade. We have compared overhead of developed algorithm with the other algorithms available in literature like min-min, SJF and FCFS. It is observed that proposed algorithm perform better than the existing algorithm because at the time of scheduling these algorithm unable to distribute the task efficiently to existing virtual machine due to which more tasks are rejected (unable to meet deadline). It is because more number of virtual machine is in overloaded and underloaded conditions hence SLA violation is increased.

7. Conclusion and future work

In this paper, we have modified the architecture of cloud resource broker and developed an efficient dynamic algorithm for task scheduling, which is based on the last optimal k-interval that not only minimizes the makespan time of tasks but also increase the ratio of tasks to meet the deadline and fulfil the objective of elasticity in cloud environment. The algorithm has been tested at variable number of task to achieve better scalability. Further, the algorithm has also been tested with modified architecture of cloud resource broker and a test scenario has been created in cloudsim. It has been observed that the developed algorithm is helpful for making intelligent scheduling decision for increasing (scale-out) or decreasing (scale in) the virtual machine instance based on the upcoming workload request/application request. The main idea of our threshold-based dynamic resource allocation scheme is to monitor and predict the resources based on the needs of the cloud applications. The performance of the reported algorithm starts to degrade when the value of last interval ‘k’ is more than 15. Experimental results show that under all possible conditions, the algorithm improves the makespan time and also number of tasks to meets the deadline. The results have proved that the developed algorithm provide better elasticity and reduce the rejection ratio of task in comparison to the existing conventional algorithms like FCFS, SJF and min-min as shown in Figs. 5–10. This model can be extended to improve other QoS parameters like the cost for ensuring the high-priority requests. The proposed algorithm can also be tested in future by using a private cloud like the OpenNebula.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compeleceng.2017.11.018.

References

[1] Hwang K, Shi Y, Bai X. Scale-out vs. scale-up techniques for cloud performance and productivity. Cloud computing technology and science (CloudCom), 2014 IEEE 6th International conference on. IEEE; 2014.
[2] Suresh A, Vijayakarthick P. Improving scheduling of backfill algorithms using balanced spiral method for cloud metascheduler. In: International conference on recent trends in information technology; 2011. p. 624–7.
[3] Dubey K, Kumar M, Chandra M. A priority based job scheduling algorithm using IBA and EASY algorithm for cloud metaschedular. In: International conference on advances in computer engineering and applications; 2015. p. 66–70.
[4] Shaoo B, Kumar D, Jena SK. Analysing the impact of heterogeneity with greedy resource allocation algorithms for dynamic load balancing in heterogeneous distributed computing system. Int J Comput Appl 2013;62(19):25–34 Jan..
[5] Li Wenzheng, Hongyan Shi. Dynamic load balancing algorithm based on FCFS. Innovative computing, information and control (ICICIC), 2009 Fourth international conference on. IEEE; 2009.
[6] Mondal RK, Nandi E, Sarddar D. Load balancing scheduling with shortest load first. Int J Comput Sci Inf Technol Res 2015;3(4):162–6 ISSN 2348-120X (online)Month: October - December.
[7] Chen H, Wang F, Helian N, Akanmu G. User-priority guided min-min scheduling algorithm for load balancing in cloud computing. Parallel computing technologies (PARCOMPTECH), 2013 national conference on; 2013.
[8] Babu D, Venkata P. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl Soft Comput 2013;13(5):2292–303 May.
[9] Ramezani F, Khadeer hussain F. Task-based system load balancing in cloud computing using particle swarm optimization. Int J Parallel Prog 2013;42(5):739–54 Oct..
[10] Pacini E, Mateos C, Garino CG. Balancing throughput and response time in online scientific clouds via ant colony optimization. (SP2013/2013/00006). Adv Eng Softw 2015;84(1):31–47.
[11] Tsai J, Fang J, Chou J. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. Comput Oper Res Elsevier 2013;40(12):3045–55.
[12] Coninck ED, Verbelen T, Vankeirsbilck B, Bohez S, Simoens P. Dynamic auto-scaling and scheduling of deadline constrained service workloads on IaaS clouds. J Syst Softw 2016;118:101–14.
[13] Naha RK, Othman M. Brokering and load-balancing mechanism in the cloud–revisited. IETE Tech Rev 2014;31(4):271–6.
[14] Somasundaram T, Govindarajan K, Rajagopalan M, Rao SM. A broker based architecture for adaptive load balancing and elastic resource provisioning and deprovisioning in multi-tenant based cloud environments. In: Advances in intelligent systems and computing, 174. Heidelberg, Germany: Springer; 2013. p. 561–73.
[15] Fu X, Chen Z. Virtual machine selection and placement for dynamic consolidation in cloud computing environment. *Front Comput Sci* 2015;9(2):322–30.
[16] Kong W, Lei Y, Ma J. Virtual machine resource scheduling algorithm for cloud computing based on auction mechanism. Optik-Int J Light Electron Opt 2016;127(12):5099–104.
[17] Abrishami S, Naghibzadeh M. Deadline-constrained workflow scheduling in software as a service cloud. Scientia Iranica 2012;19(3):680–9.
[18] Nayak SC, Tripathy C. Deadline sensitive lease scheduling in cloud computing environment using AHP. J King Saud Univ Comput Inf Sci 2016.
[19] Malawski M, Juve G, Deelman E, Nabrzyski J. Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. In: Proceedings of the international conference on high performance computing, networking, storage and analysis. IEEE Computer Society Press; 2012.
[20] Lorido-Botran T., Miguel-Alonso J., Lozano J.A., “Comparison of auto-scaling techniques for cloud environments,” (2013).
[21] Li X, Cai Z. Elastic resource provisioning for cloud workflow applications. IEEE Transactions on Automation Science and Engineering; 2015.
[22] Calheiros RN, Ranjan R, Buyya R. Virtual machine provisioning based on analytical performance and QoS in cloud computing environments. Parallel processing (ICPP), 2011 international conference on. IEEE; 2011.
[23] Righi RDR, Rodrigues VF, Rostirolla G, Costa CAD, Roloff E, Navaux POA. "A lightweight plug-and-play elasticity service for self-organizing resource provisioning on parallel applications. Future Gener Comput Syst 2017.
[24] Ghobaei-Arani M, Jabbehdari S, Pourmina MA. An autonomic approach for resource provisioning of cloud services. Cluster Comput 2016;19(3):1017–36 2017.
[25] Sindhu HS. Comparative analysis of scheduling algorithms of Cloudsim in cloud computing. Int J Comput Appl 2014;97(16).

Mohit Kumar is currently Ph.D. student at IIT Roorkee, India. He has received M.Tech from ABV-IIIT Gwalior in 2013 and B.Tech from MJP Rohilkhand University Bareilly in 2009. His research activity is mainly focus at energy efficiency, elasticity, load balancing and security in cloud Computing. He has published 8 papers in journals and international conferences.

S.C. Sharma is a professor at IIT Roorkee, India. He has published over two hundred research papers in national and international journals/conferences and supervised more than 30 projects/dissertation of PG students. He has supervised 15 Ph.D. in the area of Computer Networking, Wireless Network, Cloud Computing, Computer Communication and continuing supervising PhD in the same area.

Please cite this article as: M. Kumar, S.C. Sharma, Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment, Computers and Electrical Engineering (2017), https://doi.org/10.1016/j.compeleceng.2017.11.018