Rasim M. Alguliyev[1], Yadigar N. İmamverdiyev[2], DOI: 10.25045/jpit.v08.i2.01
Fargana C. Abdullayeva[3]
Institute of Information Technology of ANAS, Baku, Azerbaijan
[1]rasim@science.az, [2]yadigar@iit.science.az, [3]farqana@iit.ab.az

MULTICRITERIA OPTIMIZATION METHOD FOR LOAD BALANCING IN CLOUD COMPUTING

Optimizing of the task scheduling process in the cloud environment is a multicriteria NP-hard problem. In this paper, weighted load balancing method ($\alpha PSO - TBLB$) based on PSO algorithm is proposed. The method provides optimal migration of tasks from the loaded virtual machines to the less loaded virtual machine to prevent the excessive load in virtual machines of the cloud infrastructure. In the proposed optimization method, the minimization of the processing time of tasks and the transfer time of tasks were selected as the target functions. Experimental testing of the proposed approach was carried out in the Jswarm and Cloudsim programs. As a result of the simulation on the basis of the proposed method, an optimal solution for task scheduling was found, uniform distribution of tasks in virtual machines (VMs) was provided. Moreover, in the process of assigning tasks to virtual machines, a minimal time consumption was achieved.

Keywords: *cloud computing, Particle Swarm Optimization (PSO), virtual machine migration, task sheduling, Cloudsim, Jswarm, data intensive, computing intensive.*

Introduction

The cloud data center is a collection of networked servers consisting of a large range of heterogeneous hosts equipped with computational resources [1]. In the cloud data center, the virtualization technology eliminates the server heterogeneity, provides the server consolidation and improves the server efficiency [2]. The virtualized host can provide a host of multiple VMs with different workloads over time and a variety of technical resources. The servers that host the heterogeneous VMs, the workloads of which are unexpectedly changeable, may cause resource imbalance within the host. This leads to the decline in performance. As a result, the service level leads to the breach of the agreement.

To ensure the load balance in cloud technologies, VMs are migrating between the servers. VMs migration is ensured by transferring the working VMs from one physical machine to another. In this case, they gain the indicators such as computing power, memory enhancement, fast communication, and energy saving. This is a traditional approach to ensuring the system's load balance based on migration of loaded VMs in the cloud environment. This approach creates the problems such as dirty memory, capturing a large capacity both in physical machine and host physical machine, the delays in the VMs work, the time consuming and high cost migration process, and so forth. In order to eliminate these problems of the traditional approaches, instead of VMs migration, the migration of the load-generating tasks to the low-loaded VMs is carried out [3]. In the cloud environment, two groups of tasks are distinguished [4]: tasks requiring computing intensive; tasks requiring data intensity. The number of data migration is attempted to be reduced in the task scheduling requiring data intensity. Therefore, the data transmission time is reduced. In the course of task scheduling requiring computing intensity, the scheduling strategy migrates the data to high-performance computers, thereby reducing the task processing time. In cloud environment, VMs is an isolated working environment with own operating system, memory and software. Many studies were carried out to provide load balancing in the cloud environment, though the optimal solutions haven't been achieved for the distribution of additional tasks from the loaded VMs to others. The task scheduling is NP-hard and uses holographic algorithms to solve this problem [5]. This article ensures the task scheduling by using PSO algorithm.

The main idea of the proposed method is to migrate the tasks requiring computing intense to the high-performance VMs. Additionally, since the existing algorithms have the same weight of the criteria to be minimized in the scheduling process, it is not possible to adjust the target function to be more optimal. Therefore, the appropriate weight coefficients should be provided to show the

importance rate of the criteria in the scheduling process. Here, the weight coefficients are used to indicate the importance of the criteria in the scheduling process. Assimilating the weights to the criteria may provide more optimal solutions. The article aims to demonstrate that better optimal solution is obtained when assimilating the weights. Another important approach presented in the paper is to implement the *workflow scheduling*. For this purpose, the workflow is presented as a *Directed Acyclic Graph*. Here, the vertices of the graph denote the tasks, and the edges between the appropriate vertices - the dependence between the tasks. A method that provides the migration of the additional tasks to the new host VMs, using the PSO algorithm has been offered [3]. It suggests an approach similar to our proposed approach. However, conducted experiments show that there is a considerable misbalance in the load distribution. In addition, the optimization of the tasks scheduling has not been performed very successfully, and finding the optimal solution is time consuming.

This paper offers a multi-criterion optimization method, αPSO-TBLB based on the weighted PSO algorithm to eliminate the above-mentioned task scheduling problems (minimizing the time spent on scheduling, equal distribution of load between VMs, migration of dependent tasks).

The article offers a multi-criterion optimization method, αPSO-TBLB, based on weighted PSO algorithm to eliminate the above task planning problems (minimizing the planning time, equally distributed between VMs, suspension of dependent tasks). The method uses the minimization of the processing time and the transmission time of the tasks as the optimization criteria. The key innovations of the study are:

- In the cloud environment, the weighted scheduling method is suggested. Here, the optimization is ensured by minimizing the two criteria. The method focuses on minimizing the processing time and transmission time of the task.
- Based on the proposed method, the PSO algorithm is developed for task scheduling.
- The capabilities of the proposed methods are evaluated in Jswarm and Cloudsim software packages.

Types and objectives of the scheduling methods for load balancing in the cloud

Scheduling theory began to be studied in the 50s of the 20th century. Scheduling theory studies the issues that are needed to set up the sequence (configuration) of the sets of tasks. The configuration is of a general nature. These types of issues arise in the areas in which it is necessary to select the sequence in the implementation of the tasks (for example, the tasks' distribution in the production, setting up a plane landing schedule, setting up a train movement scheduling, delivering the customers services, etc.).

A *schedule* is a function that assigns a task for each machine i and during the time t. This function has various description forms [6].

The description forms of the schedule also differ. The schedules can be described in the following ways: formulas, tables, and Gantt chart. The $\alpha/\beta/\gamma$ writing, which consists of three parts, is used to describe the problem of setting up an optimal schedule. Here, α – denotes the service system, β - the characteristics of the task, γ - optimization criteria. The task scheduling and load balancing in the cloud environment is defined as follows: [7, 8]

Task Scheduling –the optimization of the tasks assigned to VMs.

Load balancing - any operation that distributes computing load between the resources.

Load Balancing - a system that distributes the customer requests across multiple servers.

Scheduling methods are divided into the following categories [9, 10]:

- *User level*. The developed method resolves the problem that occurred during the service delivery between the cloud provider and the user.
- *System level*. Provides the resource management in the cloud data center.
- *Static scheduling* .The processing of the tasks is predefined. Here, the time and sequence of each task is specified beforehand.
- *Dynamic scheduling*. The time of the task entry is unknown.
- *Centralized scheduling* uses the central processor that incorporates a set of tasks. This

device provides the tasks' transmission to other processing facilities.

- *Distributed scheduling* uses a local planning device to handle the queries and tasks. The effectiveness of this method compared to the centralized planning method is low.
- *Preemptive scheduling*. Suspension of the task processing and its migration to another resource is available.
- *Non-preemptive scheduling*. Once the task is completed, the resource that processes it may start another processing.
- *Online scheduling*. Each task is scheduled only once and the schedule outcome is unchangeable.
- *Offline scheduling*. The input tasks are not assigned to the resources, but collected and evaluated to be defined.
- *Task-level scheduling* optimizes the process of assigning the tasks to VMs at local cloud data centers. It aims to minimize all operational costs of the workflow relative to QoS (Quality of Service) indicators.
- *Service-level scheduling* performs a task assignment. The workflow tasks are assigned based on their functional and non-functional QoS requirements.

In cloud technology, load balancing approaches are mainly focusing on the provision of the following criteria [10-13]:

- *Budget*. Expenditures of the customers spent on the cloud resources they use.
- *Deadline*. The time limit for the workflow processing.
- *Reliability*. The probability of the task to be completed successfully. Scheduling methods often use the replication and backup to meet this requirement.
- *Availability*. Achieved by ensuring the cloud resource accessibility.
- *Minimizing the makespan*. The time allocated to the completion of the last task processing of the workflow.
- Service Level Agreement (SLA). Provides the QoS requirements adopted between the customer and provider.
- *Security* ensures the safe scheduling and capable to avoid the cloud-inherent security attacks.
- *Load balancing* optimizes the use of the resource by eliminating the overload of any cloud resource.
- *Response time* measures the total time spent to process the query received by the system. This time is tried to be minimized.
- *Scalability* defines the capability of the system with limited number of processors and machines to perform the load balancing algorithms.
- *Resource utilization* determines the extent to which the system uses resources. The best load balancing algorithm should provide the maximum use of resources.
- *Migration time*. The time spent to migrate the task from one machine to another in the cloud system. To improve the performance of the cloud system, this indicator needs to be minimized.
- *Performance* shows the effectiveness of the system after the load is balanced. It is assumed that the system's performance may increase if the above parameters are fully met.
- *Throughput*. The total number of the tasks. The higher this parameter is, the more effective the system is considered to be.
- *Associated overhead*. The total cost of performing the load balancing algorithm. Achieving the minimum cost is an indication that the algorithm is more successful.
- *Fault tolerant*. The ability of the algorithm to run precisely at any system fault.

Related studies

Developed algorithms for task scheduling and load balancing significantly reduce the number of SLA failures in cloud environment. Load balancing algorithms are divided into dynamic and static categories. Static algorithms do not depend on the current state of the system, and the precondition is to be aware of the state of the system in advance. Dynamic algorithms are related to the current state of the system. Static algorithms can work correctly if there is a few changes are made to the node load. Static algorithms are not considered functional for cloud environment since the load changes in this environment occur at different times. Therefore, the scientific community does not focus on the development of the static algorithms for load balancing in large computing environments such as cloud. Cloud technologies need the optimal load balancing mechanism. In this regard, this section of the study provides an overview of the dynamic load balancing algorithms.

Dynamic Load Balancing Algorithms. One of the methods ensuring the load balancing in the cloud environment is migrating the tasks to VMs. This is ensured through the migration of the load-generating tasks to less loaded VMs [3]. [14] offers a comprehensive multi-purpose optimization method for task scheduling through minimizing the processing time, transmission time and processing costs of the task. The developed optimization method uses contradictory target functions and finds the optimal solution through the PSO algorithm. [15] offers an optimization method that migrates the load-generating tasks into new VMs by applying the PSO algorithm. The method takes the minimization of the processing time and transmission time of the task as the key optimization criteria. The offered method also provides the unused memory capacity, memory usage and cost reduction, as the VMs do not stop during the task migration. The disadvantage of the method is that it migrates the only tasks entered into the loaded VMs, but does not depend on each other into the other homogeneous VMs. [16] proposes a multi-criteria optimization method based on genetic algorithm for task scheduling, taking into account the transmission time and processing costs of the tasks, energy consumption, length of the tasks' sequence as the basic minimization criteria. The four objectives of the method based on the criteria listed are contradictory. As the minimization of the processing and energy expenditures is the key target here, the method reduces the costs both for the customer and the provider. Since the target functions contradict to one-another, the method also reduces the respond time and the makespan. The disadvantage of the proposed method is that it spends much time on the task scheduling process and it is slow. [17] offers a dynamic scheduling algorithm to balance the task burden between the heterogeneous resources. In this approach, the load balancing in the cloud environment is modeled inspired by the behavior of bees in the search for food source. In this regard, bee algorithm is used to provide the load balancing between VMs. Here, the bees are modeled as the tasks to be balanced, VMs -as the source of food, and less loaded machines - the destination of the bees. When the VM is loaded, the task is directed to the less-loaded VM. The time spent on waiting for the VM is attempted to be minimized and the bandwidth to be increased. Taking non-preemptive scheduling as a key requirement, it greatly reduces the response time and the makespan. The advantage of the proposed method is that it balances the load in cloud environment and takes into account the sequence priority of the task waiting for the VM. Migrating the inter-independent tasks for the load balancing and not being sufficiently scalable are the main disadvantages of the method. One of the modeled approaches to the load balancing between the VMs inspired by the honeybee feeding behavior is proposed in [18]. The tasks taken from the loaded VMs are considered as honey and low loaded VMs –as the food source. The proposed method consists of four blocks. The first block calculated the current workload of the VM, the second block makes a decision for load balancing and scheduling, the third block groups the VMs, and the fourth block performs the task scheduling. Basically, this method tries to minimize the makespan and the number of VM migrations. The disadvantage of the method is its low scalability. The efficient organization of load balancing, optimization of response time, improving the task implementation time and makespan, reducing the inefficiency in bandwidth are the criteria affecting the QoS in cloud environment. [19] offers a new strategy for load balancing. Based on the strategy, the tasks taken from the task list are placed in VMs with high computing power. The method aims to improve the QoS of the cloud by optimizing the maximum load of VMs and the implementation

time of the task. The weakness of method is that it is not scalable. [4] offers a method for task scheduling, based on PSO algorithm, built on *small position value (SPV)* to minimize processing costs and transmission time. The method is very useful in heterogeneous environments. The disadvantage of the method is that it does not provide scalability. The properties and parameters of the dynamic load balancing algorithms described above are presented in Table 1.

Many hybrid algorithms have been proposed for load balancing in cloud computing [20, 21]. As the presented article does not suggest the hybrid approach to dynamic load balancing, the existing hybrid algorithms are not analyzed here.

Table 1

Dynamic load balancing algorithms and their parameters

The method used	Idea	Reference	Advantage	Disadvantage
PSO algorithm based on SPV	migration of the tasks requiring computing intense to a high performance computer	[4]	• minimizing processing costs; • minimizing transmission time; • favorable for heterogeneous systems.	scalability not provided
PSO	assigning the load-generating tasks to appropriate VMs	[14] [15]	• minimizing makespan • minimizing task transmission time	inter-independent tasks
genetic algorithm	assigning the load-generating tasks to appropriate VMs	[16]	• task transmission time; • task processing costs ; • energy consumption; • length of the task queue.	time consuming and slow task scheduling
bee algorithm	models the feeding behavior of bees	[17]	• minimizing the task waiting time for the VM; • increasing the bandwidth; • considering the priority of the task sequence waiting for VM; • minimizing the response time; • minimizing the makespan	• migrates inter-independent tasks; • lack of scalability
modified honey-bee algorithm	models the feeding behavior of honey-bees	[18]	• minimizing the response time; • minimizing the task implementation time; • minimizing the makespan; • minimizing the number of VM migrations.	low scalability
a new strategy for task deployment	providing the load balancing using task deployment strategy	[19]	• minimizing the maximal load of VM; • minimizing the task completion time.	not scalable

Scheduling problem statement

The article provides a method for workflow tasks' scheduling. The workflow is often described in the form of a *Directed Acyclic Graph (DAG)* and written as $G = (V,\ E)$ [4]. Here $V = \{T_1,\ T_2, \dots ,\ T_n\}$ – the tasks in the workflow, n - the total number of the tasks in the workflow. $E = \{d_{ij}\},\ 1 \le i,\ j \le n$ is the sets of arcs, and indicates the dependence of the data between the tasks. Here, the arc $d_{ij} = (T_i,\ T_j) \epsilon E$ indicates that the data is transmitted from the task $T_i$to the task T_j.

Assume that there is *m* number of VMs in cloud environment: $VM = \{VM_1,\ VM_2, \dots ,\ VM_m\}$. According to the proposed optimization function for scheduling, it is required to find the optimal solution for the assignment of the tasks to VMs. Here, the task execution time T_{exe} and minimization of the task transmission time (T_{trans}) are the target functions. These criteria are calculated as follows:

$$T_{exe}=\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ik} * \frac{DE_i}{VM_{m_k}} = \frac{[number\ of\ transactions]}{number\ of\ transactions/sec.} = \text{sec} \quad (1)$$

Where DE_i- the workload of the *i*-th task (indicated by the number of transactions); VM_{m_k}- speed of the processors at the *k*-th VM (operation/sec); m – the number of VMs; n – the number of tasks.

$$T_{trans}=\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{z=1}^{m}(1-x_{ik}*x_{iz})\frac{DT_{kz}}{B_{kz}} \quad (2)$$

where DT_{kz}- the volume of data exchanged between virtual machines *k* -th and z -th. It is equal to the volume of the file(s) of the *i*-th task (bits); B_{kz}, $k,z=\{1, 2, \ldots, m\}$- the bandwidth between two VMs (bit/sec), $x_{iz}=1$ i -th task is taken from the *k* -th VM and assigned to the z-th VM and, in this case, $x_{ik}=0$.

The main problem of the existing methods is the choice of criteria. The existing algorithms have the same weight of the criteria to be minimized in the scheduling process. Relevant weight coefficients should be given to show the significance degree of the criteria in the scheduling process. Here, the weight coefficients are used to indicate the significance of criteria in the scheduling process. Assimilating the weights to the criteria can provide better optimal solutions. This relation is shown as follows:

$$U=\alpha * T_{exe} + (1-\alpha) * T_{trans} \rightarrow min \quad (3)$$

Where α is the processing time of the task, and the $(\alpha \in (0,1))$, $(1-\alpha)$ is the weight coefficient given to the task transmission time. Here, it is possible to easily adjust the target function to be more optimal by changing the value α.

Limitations

$$\sum_{k=1}^{m} x_{ik}=1, \quad \forall i=1,\ldots,n \quad (4)$$

where $x_{ik}=1$,when the i -th task is assigned to the *k*-th VM, otherwise $x_{ik}=0$.

$$\sum_{k=1}^{m}\sum_{z=1}^{m}(1-x_{ik}\times x_{iz})=1, \quad \forall i=1,2,\ldots,n, \quad k\neq z \quad (5)$$

$$x_{ik},(1-x_{ik}*x_{iz})\in\{0,1\},\forall i,k,z$$

where the i -th task is taken from the *k*-th VM, and if it is assigned to the *z*-th VM, $x_{iz}=1$, otherwise $x_{iz}=0$.

The following variables are used in the method:

$T_{set}=\{T_1,T_2,\ldots,T_n\}$- the set of tasks to be migrated;

VM_j – the *j*-th virtual machine, $j=\{1,2,\ldots,m\}$;

MIPS - million instructions per second.

Canonical model of PSO algorithm

The canonical model of the PSO algorithm consists of particles. These particles "fly" in the hyper-dimensional search field. Position changes of the particle in the search field are based on the socio-psychological tendency of the individuals. Here, the position of each particle varies according to its own and neighbors' skills.

The particles in the PSO algorithm are presented as candidate solutions. The particles perform the iterative steps to find a solution of the target function with the best value. Each particle has the position depicted by the position vector $\vec{x}_i$ (i - particle's index), and the velocity described by the velocity vector $\vec{v}_i$. In the iterative process, each particle memorizes the best position in the vector $\vec{x}_{pbest_i}$. The best position vector within the colony is stored in the vector $\vec{x}_{gbest_i}$. The velocity vector is updated at each iteration t using the formula (6). Subsequently, a new position is formed by summing a new velocity and the previous position based on formula (7).

$$\vec{V_i}(t+1)=W\vec{V_i}(t)+C_1r_1\left(\vec{x}_{pbest_i}-\vec{X_i}(t)\right)+C_2r_2\left(\vec{x}_{gbest_i}-\vec{X_i}(t)\right) \quad (6)$$

$$\vec{X_i}(t+1)=\vec{X_i}(t)+\vec{V_i}(t+1) \quad (7)$$

where $\boldsymbol{W}$ – is an inertia factor $\boldsymbol{W = 0.95}$, $\boldsymbol{C_1}$ - a positive integer and declares itself as a recognition coefficient $\boldsymbol{C_1 = 0.8}$, $\boldsymbol{C_2}$ – a positive integer and declares itself as a social component coefficient ($\boldsymbol{C_2 = 0.8}$). Here must be $\boldsymbol{C_1 + C_2 \leq 4}$. $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are the random numbers used to represent the population's differences and take values in the range [0, 1].

Here, according to the formula (6), the particle determines in which position it will move next time based on its position and the position of the most successful particle in the colony.

All the particles' positions generated using PSO (6) and (7) formulas are produced by a certain vector. The values of this vector are uninterruptable by nature [22], and since it is impossible to determine the number of VMs with these continuous values, it is important to convert them into discrete units [22, 23].

The small position value is used to convert the vector $x_i^k = [x_1^k, x_2^k, \dots, x_n^k]$, which consists of continuous values of the particles' position, into the discrete vector [4, 24].

Small Position Value

To convert the continuous position vector $x_i^k = [x_1^k, x_2^k, \dots, x_n^k]$, into the vector $s_i^k = [s_1^k, s_2^k, \dots, s_n^k]$, which consists of discrete units, the SPV procedure is as follows [4, 24].

According to the SPV, all the tasks, including those with the smallest PSO value in the solution list x_i^k , are numbered for their incremental sequence and included into the field s_i^k (Table 2). Here, $i -$ is the i-th particle (PSO solution), $k -$k-th iteration.

Table 2

Discretization of continuous PSO values through the SPV procedure

Tasks	1	2	3	4	5	6	7	8	9	10
x_i^k	0.1587	3.6189	2.3824	0.0292	0.8254	0.4256	0.3679	0.1276	0.6378	0.7832
s_i^k	3	10	9	1	8	5	4	2	6	7

As seen from Table 2, the task schedule after SPV conversion is 3-10-9-1-8-5-4-2-6-7.

Once the SPV values are formed, the elements of the conversion vector s_i^k are adjusted to the appropriate VM vector with the following formula, and thus, the scheduling vector is formed:

$$p_i^k = s_i^k \; mod \; m + 1 \quad (8)$$

where, m represents the total number of virtual machines. In our case, 5 VMs are taken (VM_i^k= 1, 2, 3, 4, 5) x_i^k- the particle vector consisting of continuous values derived through PSO. s_i^k - discrete values obtained as a result of SPV conversion of continuous values x_i^k. Adjusting the discrete values (tasks) s_i^k to the VM through the formula (8) is as follows:

$$p_1 = 3 \; mod \; 5 + 1 = 3 + 1 = 4;$$
$$p_2 = 10 \; mod \; 5 + 1 = 0 + 1 = 1;$$
$$p_3 = 9 \; mod \; 5 + 1 = 4 + 1 = 5;$$
$$p_4 = 1 \; mod \; 5 + 1 = 1 + 1 = 2;$$
$$p_5 = 8 \; mod \; 5 + 1 = 3 + 1 = 4;$$
$$p_6 = 5 \; mod \; 5 + 1 = 0 + 1 = 1;$$
$$p_7 = 4 \; mod \; 5 + 1 = 4 + 1 = 5;$$
$$p_8 = 2 \; mod \; 5 + 1 = 2 + 1 = 3;$$
$$p_9 = 6 \; mod \; 5 + 1 = 1 + 1 = 2;$$
$$p_{10} = 7 \; mod \; 5 + 1 = 2 + 1 = 3.$$

Table 3

Adjusting the continuous values of PSO algorithm based on 10 tasks and 5 VMs

Number of tasks	x_i^k	s_i^k	Adjusting to the VMs (p_i^k)
1	0.1587	3	4
2	3.6189	10	1

3	2.3824	9	5
4	0.0292	1	2
5	0.8254	8	4
6	0.4256	5	1
7	0.3679	4	5
8	0.1276	2	3
9	06378	6	2
10	0.7832	7	3

PSO parameters for load balancing

The proposed method uses the PSO algorithm to provide the effective load balancing between the VMs and moves the load-generating tasks in less loaded VMs. The PSO algorithm parameters are adjusted to the cloud environment in Table 4.

Table 4

Adjusting the PSO algorithm to the cloud environment

PSO parameters	Cloud environment
Particle	Tasks
Particle positions	VM number
Optimal solution	*n*-dimensional vector consisting of *N* number tasks. For example, *cloudlet*. The elements of this vector are the tasks and take values in the range [1, n].
Particle migration	Moving the task to less-loaded VM with high power

In the proposed method, the tasks are the particles. At the initial stage, where the particles search for "food source", the tasks are assigned to VMs to be processed. As the processing capacities of different VMs differ, sometimes, these VMs are exposed to overloading while the others are less loaded. When the appropriate VM is loaded with multiple tasks, some tasks are migrated to the less-loaded VMs.

Experiment

Jswarm and Cloudsim software have been used to evaluate the capabilities of the proposed method. BindCloudletToVm () Broker of the DatacenterBroker class by Cloudism assigns tasks to VMs. However, Jswarm finds the optimal layout of the tasks between the VMs based on the PSO algorithm. In the $\alpha PSO-TBLB$ model offered in the current article, the broker bindCloudletToVm () assigns the tasks to the VM based on the Jswarm algorithm.

Three physical machines (data centers), five VMs and ten tasks are taken to create a cloud simulation environment in Cloudsim software. The data used for VMs and tasks in the $\alpha PSO-TBLB$ model are given in Table 5 and Table 6.

Table 5

Properties of VMs

Number of VM	Velocity (MIPS)	VM description size	VM memory (ram)	Bandwidth	Number of CPU	Virtual machine monitor name (VMM)
1	250	1,000	512	1,000	1	Xen
2	300	1,000	256	1,000	1	Xen
3	250	1,000	512	1,000	1	Xen
4	250	1,000	512	1,000	1	Xen
5	250	1,000	512	1,000	1	Xen

It is assumed that there are five affordable VMs to ensure the load balancing based on the

proposed method. In this case, the next step, the load-generating tasks in the loaded VM are optimally assigned to the appropriate VM using Cloudsim program package by minimizing the makespan and the transmission time of the task.

Table 6

Properties of the tasks

Number of tasks	Number of task's operations	Volume of task's file	Volume of output data of the task	Number of processors (CPUs)
1	250,000	300	300	1
2	25,000	300	300	1
3	250,000	300	300	1
4	25,000	300	300	1
5	250,000	300	300	1
6	250,000	300	300	1
7	25,000	300	300	1
8	250,000	300	300	1
9	250,000	300	300	1
10	25,000	300	300	1

The optimal schedule for scheduling these tasks over the VMs is found based on the proposed αPSO − TBLB algorithm. Later, this optimal schedule has been used for the optimal tasks scheduling by the bindCloudletToVm () broker in Cloudsim software.

[3] offers a method for migrating additional tasks to a new host VMs by applying the PSO algorithm. This approach is close to our proposed approach. However, the conducted experiments show that there is a serious imbalance in the load distribution. Here, two tasks t_5 and t_6 are assigned to the machine VM1, four tasks- t_2, t_4, t_7 and t_8–to the machine VM3, two tasks - t_1 and t_{10} - to the machine VM4, two tasks - t_3 and t_9 - to the machine VM5, while VM2 does not participate in the process. However, the result obtained in Cloudsim, based on the αPSO − TBLB algorithm (Table 7), proves that in our approach, this imbalance is seriously eliminated by applying the weight coefficient α. Thus, at the weight coefficient value α = 0.4839, the tasks are distributed equally among the VMs and all the VMs are involved in the task processing. As seen in Table 7, the machine VM1 is loaded with two tasks - t_1 and t_5, machine VM2– with one task- t_4, machine VM3– with two tasks- t_8, t_{10}, machine VM4– with two tasks - t_6 and t_9, and machine VM5– with three tasks- t_2, t_3 and t_7.

Table 7

Simulation results

Tasks	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
VMs	vm_1	vm_5	vm_5	vm_2	vm_1	vm_4	vm_5	vm_3	vm_4	vm_3
Total Task Scheduling Time: 0.211 seconds										

In addition, the overall time spent on tasks scheduling in the αPSO − TBLB method has also been very successfully optimized. Thus, in [3] this duration is 0.224 seconds, whereas in the αPSO − TBLB method this figure is 0.211 seconds (Figure 1).

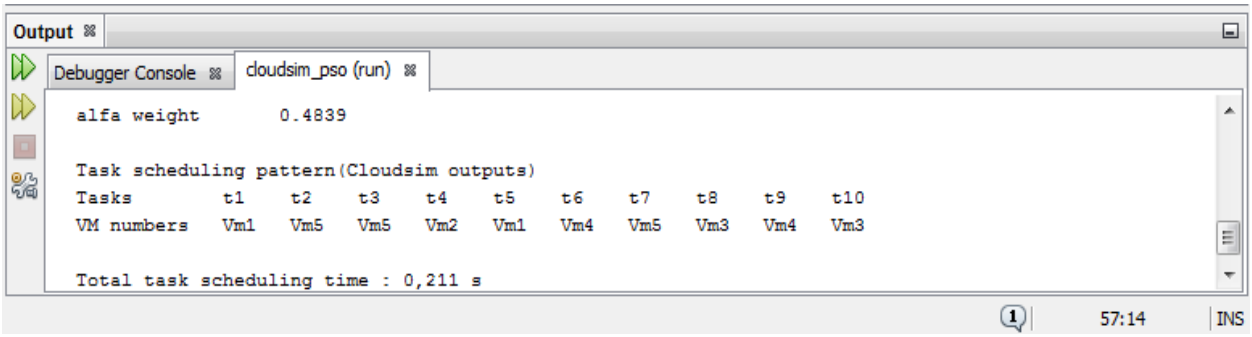

Figure 1. Task scheduling in Cloudsim

The positions of the particles depicted in Figure 1 are optimal solutions according to the PSO algorithm: $d(\overline{X_i}) = (d_1,\ d_2\ ,\ \ldots\ ,\ d_{10}) = (1,\ 5,\ 5,\ 2,\ 1,\ 4,\ 5,\ 3,\ 4,\ 3)$. According to this optimal solution, the virtual machines $vm_1, vm_5, vm_5, vm_2, vm_1, vm_4, vm_5, vm_3, vm_4, vm_3$ are selected to process the tasks $t_1,\ t_2,\ t_3,\ t_4, t_5, t_6, t_7, t_8, t_9, t_{10}$ respectively.

In order to ensure the traditional load balancing in the cloud environment, a new VM must be built at first place. In this environment, the process of setting up VM takes about 5 to 15 minutes. However, the load balancing based on the offered αPSO − TBLB model takes only 0.211 seconds. This is a significantly short period of time compared to the traditional load balancing approach.

The process of scheduling is followed by the migration process. Traditional load balancing approaches need to migrate the initial VM to the new host VM as a whole. However, the proposed αPSO − TBLB method migrates only the load-generating tasks to appropriate VMs. Evidently, migration of a whole VM takes a lot of time and memory. However, only the task migration process takes a few seconds and memory loading is almost prevented.

Conclusion

Cloud technologies are a multi-criterion environment. Cloud resource scheduling based on one criterion in this environment causes a serious load imbalance among VMs. In this regard, the article introduces a new multi-criterion optimization technique to address the weighted task scheduling problem. The key problem of the existing methods, proposed in the field of task-scheduling for load balancing, is the choice of criteria. The existing algorithms have the same weight of the criteria to be minimized in the scheduling process. Relevant weight coefficients should be given to show the significance degree of the criteria in the scheduling process. Assimilating the weights to the criteria can provide better optimal solutions. It is possible to easily adjust the target function to be more optimal by changing the value of the weight coefficients.

References

1. Metri G., Srinivasaraghavan S., ShiW., Brockmeyer M. Experimental analysis of application specific energy efficiency of datacenters with heterogeneous servers / Proc. of the IEEE 5th International Conference on Cloud Computing, 2012, pp.786−793.
2. Vaquero L.M., Rodero-Merino L., Caceres J., Lindner M. A break in the clouds: towards a cloud definition // ACM SIGCOMM Computer Communication Review, 2008, vol.39, no.1, pp.50−55.
3. Ramezani F., Lu J., Hussain F.K. Task-based system load balancing in cloud computing using Particle Swarm Optimization // International Journal of Parallel Programming, 2013, vol.42, no.5, pp.739−754.
4. Guo L., Zhao S., Shen S., Jiang C. Task scheduling optimization in cloud computing based on heuristic algorithm //Journal of Networks, 2012. vol.7, no3, pp.547−553.
5. Alguliev R.M., Alyguliev R.M., Alekperov R.K. An approach to optimal task assignment in a distributed system // Journal of Automation and Information Sciences, 2004, vol.36, no.10, pp.51–55.

6. Tanaev V.S., Gordon V.S., Shafransky Ya.M. Theory of schedules. One-stage systems. M.: Science. The main edition of physical and mathematical literature, 1984, p.384.
7. Wu Z.,Liu X., Ni Z.,Yuan D.,Yang Y. A market-oriented hierarchical scheduling strategy incloud workflow systems // The Journal of Supercomputing, 2013, vol.63, no.1, pp.256−293.
8. AjitM.,Vidya G. VM level load balancing in cloud environment / Proc. of the fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2013, pp.87−95.
9. Chawla Y., Bhonsle M. A study on scheduling methods in cloud computing // International Journal of Emerging Trends &Technology in Computer Science (IJETTCS), 2012, vol.1, no.3, pp.12−17.
10. MohammadM., ValiKardan S., Shahi Z., Azar S.I. Towards workflow scheduling in cloud computing: A comprehensive analysis // Journal of Network and Computer Applications, 2016, vol.66, pp. 64−82.
11. Milani A.S., Navimipour N.J. Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends // Journal of Network and Computer Applications, 2016, vol.71, pp.86−98.
12. Ajit M., Vidya G. VM level load balancing in cloud environment / Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2013, pp.1−5.
13. Madni S.H., Latiff M.S., Coulibaly Y., Abdulhamid S.M. Resource scheduling for Infrastructure as a Service (IaaS) in cloud computing: Challenges and opportunities // Journal of Network and Computer Applications, 2016, vol.68, pp.173−200.
14. Ramezani F., Lu J., Hussain F. Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization // Service-Oriented Computing, 2013, vol.8274, pp.237−251.
15. Ramezani F., Lu J., Hussain F.K. Task-based system load balancing in cloud computing using particle swarm optimization // Knowledge Engineering and Management, 2013, pp.31−42.
16. Ramezani F. Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments // World Wide Web, 2015, vol.18, no.6, pp.1737−1757.
17. Dhinesh B.D., Krishna P.V. Honey bee behavior inspired load balancing of tasks in cloud computing environments // Applied Soft Computing, 2013, vol.13, no.5, pp.2292−2303.
18. Babu K.R., Samuel P. Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud // Innovations in Bio-inspired Computing and Applications, 2016, pp.67−78.
19. Banerjee S., Adhikari M., Kar S.,Biswas U. Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud // Arabian Journal for Science and Engineering, 2015, vol.40, no.5, pp.1409−1425.
20. Liu Y., Zhang C., Li B., Niu J. DeMS: A hybrid scheme of task scheduling and load balancing in computing clusters // Journal of Network and Computer Applications, 2015, pp.1−8.
21. Cho K.M., Tsai P.W., Tsai C.W., Yang C.S. A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing // Neural Computing and Applications, 2015,vol.26,no.6, pp.1297−1309.
22. Alguliev R.M., Aliguliyev R.M., Mehdiyev C.A. An optimization approach to automatic generic document summarization // Computational Intelligence,2013, vol.29, no.1, pp.129−155.
23. Aliguliyev R.M. Clustering techniques and discrete particle swarm optimization algorithm for multi-document summarization // Computational Intelligence, 2010, vol.26, no.4, pp.420−448.
24. Cakar T., Koker R. Solving Single Machine Total Weighted Tardiness Problem with Unequal Release Date Using Neurohybrid Particle Swarm Optimization Approach // Computational Intelligence and Neuroscience, 2015, vol.2015, pp.1−13.