King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

Deadline sensitive lease scheduling in cloud computing environment using AHP

Suvendu Chandan Nayak [*], Chitaranjan Tripathy

Department of Computer Science and Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, India

Received 18 January 2016; revised 12 May 2016; accepted 31 May 2016
Available online 11 June 2016

KEYWORDS

OpenNebula;
Deadline sensitive;
Haizea;
Backfilling;
Scheduling;
AHP

Abstract The OpenNebula is an open source environment which provides cloud resources with the help of Haizea as a lease manager. The Haizea supports different types of leases from which deadline sensitive lease is one of them. In real time, most of the leases are deadline sensitive leases. These deadline sensitive leases are scheduled by using the backfilling algorithm. In the backfilling algorithm one of the lease is selected from the best effort queue which will provide the free resources to schedule the deadline sensitive lease. But in some scenario backfilling algorithm does not provide better scheduling if there is similar types of leases and must be in conjugative in sequence. This work aims to use AHP (Analytic Hierarchy Process) as a decision maker in the backfilling algorithm to choose the possible best lease from the given best effort queue in order to schedule the deadline sensitive lease. The proposed work improves the performance of the backfilling algorithm by scheduling more number of leases and minimizing the lease rejection using AHP.

1. Introduction

In recent days, the cloud computing has appeared as a new era of resource computing which provides resources as a *service on demand* as per the user's request. The task of the cloud service providers to provide on-demand-resources for the users is quite challenging. So, resource management is very much essential for the cloud service providers (Liu et al., 2014) for better utilization of resources to increase the system performance when workload increases. Processing resource management is carried out by resource monitoring, resource provisioning and resource allocation. According to the service level agreement (SLA), the service provider needs to fulfill the *on demand request* of the user. The User request may be sent at any time to the cloud service provider for the resources (Nathani et al., 2012) and should be processed to execute the user applications.

Many cloud enabling open source platforms are being used by the cloud service providers to provide cloud services (Liu et al., 2014) to run user applications by creating a user friendly environment and also to manage datacenters. In order to manage the cloud resources different mechanisms and algorithms are adopted. The OpenNebula is an open source software clouds toolkit which manages the data centers and clouds. A data center consists of a number of physical machines. In each

[*] Corresponding author.
E-mail addresses: Suvendu2006@gmail.com (S.C. Nayak), crt.vssut@yahoo.com (C. Tripathy).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2016.05.003

physical machine a number of virtual machines (VMs) are created according to the user's request to compute user applications. These types of requests are considered as advance reservation leases (Sotomayor et al., 2009b) which needs required resources not right now but in future to compute the user applications.

The Haizea is an open-source VM-based lease management architecture which is used in OpenNebula as a resource manager (Sotomayor et al., 2008). It accepts the user request as lease and then schedules the lease according to the availability of resources in the physical machines. Moreover, OpenNebula is an open source where as Haizea is a lease manager that acts as a scheduler in the backend for OpenNebula ("Opennebula – Flexible Enterprise Cloud Made Simple", 2016). The user requests are categorized as the following by Haizea: Best-effort lease, Advance reservation-style leases (AR) and Immediate lease (Sotomayor, 2009; "Haizea – An Open Source VM-Based Lease Manager", 2016).

Best-effort lease

Resources are allocated to the lease as soon as they are available. The possibility of resource allocation is service provider dependant. The request for resource is processed by the scheduler when the resources are free. No time constraints are associated with this type of lease.

Advance reservation-style leases (AR)

The AR lease is another type of lease which requires resources within a strictly determined time period. In AR lease as the name suggests, the service provider provides resources in the predetermined time period according to the user's request otherwise the SLA is violated.

Immediate leases

In this lease, resources must be allocated either instantly, or not at all. These types of leases need to use the resources at an instance. The scheduler processes these leases if resources are available, otherwise not.

Besides these leases another type of lease is possible with time constraint called "Deadline sensitive lease" (Nathani et al., 2012) in OpenNebula. To schedule deadline sensitive leases swapping and backfilling algorithms are used in Haizea. The scheduler selects a lease from the best effort queue which will provide free slots to the schedule deadline sensitive lease. A decision maker can be utilized to select the lease.

The AHP is an effective tool to handle the complex decision making problems (Saaty, 2008). It uses a set of evaluation criteria, and a set of alternative options to make a best decision. The rank is found out by using *pairwise comparison* (Saaty, 2003) among the criterions. The AHP has a wide verity of applications like strategic planning, resource allocation, source selection, business or public policy, program selection and many more. The task scheduling is done by AHP where the AHP is used to find the priority of the task (Ergu et al., 2011a,b).

In this proposed work, we improved the scheduling performance of the backfilling algorithm by using AHP. The AHP is used as a decision maker to select the lease from the best effort queue which provides free slots to allocate deadline sensitive lease. In Section 2, the previous works related to the proposed works are discussed. The Existing scheduling mechanisms are discussed in Section 3. The proposed scheduling mechanism using AHP with illustration is presented in Section 4. Sections 5 and 6 followed the result analysis and conclusion along with future work of the proposed work.

2. Related work

The Elasticity is an important characteristic of cloud computing. It provides a scale up and scale down of the on demand resources for the user (Li and Cai, 2015). For which the author proposed a task depth based priority rule to schedule the current task. The processes are used to manage physical resources such as CPU cores, Main memory, disk space, I/O devices and network bandwidth. These resources must be sliced and shared between virtual machines running potentially heterogeneous workloads (Manimaran and Murthy, 1998).

The Resource allocation is a very challenge area in all types of computing like distributed computing, parallel computing, grid computing, green computing and cloud computing. The Resource allocation in cloud computing has attracted significant attentions of researchers. Resource allocation is a NP hard problem. The Bin packing algorithm is used to allocate each task to a processor (Manimaran and Murthy, 1998); (López et al., 2003).Their work is based upon periodic task with heterogeneous resources in grid computing. For resource scheduling in grid computing different scheduling algorithms have been proposed. In the literatures, these are based upon policies, objective functions, applications models, adaptation, QoS constraints for static and dynamic environment (Dong and Akl, 2006).

The recent researches in cloud environment have focused on various resource allocation algorithms that are used to schedule user requests. A cost-optimized and deadline-constrained algorithm was proposed by Byun et al. (2011). Their work uses only one type of cloud resource. The Partitioned Balanced Time Scheduling (PBTS) takes scheduling decision for user applications and provides resource provisioning. Moreover, Abrishami et al. (2013) proposed an algorithm by considering cost and deadline as constraint for user application in cloud environments.

Different optimization techniques were also proposed by researchers for resource allocation in clouds. A PSO based algorithm is proposed by Pandey et al. (2010) to minimize the execution cost of the task. The proposed mechanism provides load balancing for the available resources. Another PSO based work was proposed by Wu et al. (2010) for finding a near-optimal scheduling. The work focuses on minimizing cost or time by considering the deadline for the task. Moreover a review and comparison are studied using different popular metaheuristic techniques: Ant Colony Optimization (ACO), Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), and two novel techniques: League Championship Algorithm (LCA) and BAT algorithm for cloud and grid environment (Kalra and Singh, 2015). Rodriguez and Buyya (2014) proposed resource provisioning and scheduling in IaaS (Infrastructure as a Service) using PSO. These tasks are considered with a time constraint called deadline. Calheiros and Buyya (2014) proposed task migration based on deadline. Free time slots are found and the tasks are fit in these time slots by the proposed algorithm, otherwise tasks are migrated to other VM. The migration and scheduling of tasks also meet the deadline.

Most of the works proposed in the past are based upon resource allocation or task scheduling for similar types of task. Sotomayor and Keahey (2007) also focused on the schedule resources required by a task for a short period of time which is called on-demand short-term lease. The architecture suggested by them is cost effective for service providers by scheduling short-term leases along with existing workloads. Moreover, Sotomayor et al. (2008) proposed load scheduling by suspending low priority lease in Haizea for OpenNebula platform. Recently immediate and, best effort resource allocation policies are mostly used by cloud service providers for resources to process user request in IaaS (Sotomayor et al., 2009b). The three different types of leases that are supported by Haizea were also discussed. Similarly, another model was proposed by Sotomayor et al. (2009a,b) for predicting various run-time overheads involved in virtual machines for the AR leases using scheduling decisions. The scheduling decisions were enacted by using Haizea with the OpenNebula.

To schedule best-effort leases, the backfilling algorithm is used. The backfilling algorithm is most commonly optimized scheduling algorithm (Lifka, 1995; Feitelson, 1998). Nathani et al. (2012) proposed a mechanism to schedule deadline sensitive leases using swapping and backfilling algorithm. Ergu et al. (2011a,b) scheduled tasks using AHP in cloud where, the AHP is used to find the priority of the tasks. But in real time, we need some decision support system in Haizea with OpenNebula to schedule deadline sensitive leases using backfilling. The problem related to backfilling and the solutions using AHP is presented in Sections 4 and 5.

3. Scheduling mechanisms

This section discusses the various scheduling mechanisms. OpenNebula is more popular for virtual infrastructure manager as compared to other cloud open source platform (Sotomayor et al., 2009b). In the back end, Haizea is used as a resource lease manager for OpenNebula. Haizea accepts four types of leases like Best-effort lease, advance reservation lease, Immediate leases and Deadline sensitive lease (Nathani et al., 2012). The Deadline sensitive leases are best-effort leases with a certain time limit. When a user request needs to get resources for best-effort leases within a certain time limit, these types of leases can be considered as deadline sensitive lease in Haizea. The Deadline sensitive leases are preemptive in nature if and only if it meets its deadline. The swapping and backfilling are used to schedule these types of leases in Haizea (Nathani et al., 2012). The backfilling algorithm is used when swapping fails to schedule the deadline sensitive leases.

Example 1. The first example illustrates how swapping algorithm fails to schedule the deadline sensitive leases in Haizea. Table 1 describes the deadline sensitive lease information which consists of nodes, submit time, start time, duration (execution) time and finish time (deadline) of each lease.

By using the swapping algorithm (Nathani et al., 2012) two consecutive leases can be swapped if and only if, the first lease has requested fewer resources than the second lease and after swapping, they must finish their execution within their deadline. So the lease 1(L1) and lease 2(L2) should be swapped, though lease 2 requires 3 nodes. Therefore, the leases 2 will be scheduled before lease 1 for the duration of 40 min its execution time where lease 1 is unable to meet its deadline. Similarly Lease 3(L3) and lease 4(L4) are scheduled according to their execution time with the time slots (12.40–1.30 PM) and (1.30–1.50 PM) respectively to meet their deadline as shown in Fig 1.

Table 1 Lease information (Nathani et al., 2012).

Lease no.	Nodes	Submit time (AM)	Start time (PM)	Duration	Deadline (PM)
1	2	11.10	12.00	20	12.30
2	3	11.20	12.00	40	01.00
3	2	11.30	12.00	50	01.50
4	4	11.40	01.00	20	01.50

The swapping algorithm produces better resource allocation than simple allocation policy (Nathani et al., 2012). But in some scenarios, the swapping algorithm is unable to schedule the leases. When swapping fails, the backfilling is used to schedule the leases. The Fig. 2 shows the scheduling of leases of Table 1 using backfilling algorithm.

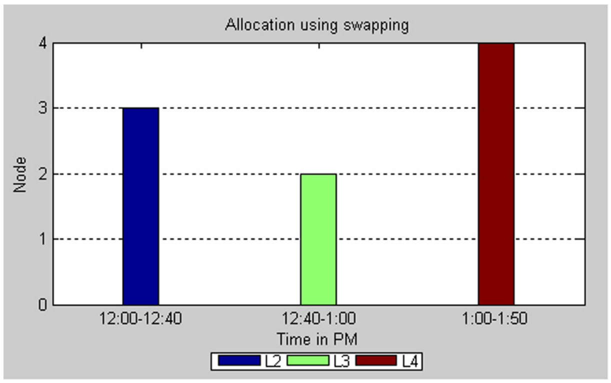

Figure 1 Lease scheduled using swapping.

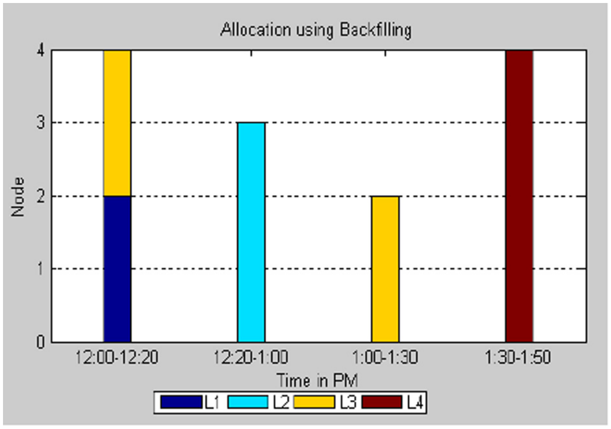

Figure 2 Lease scheduled using Backfilling.

In Fig. 2, all the leases are scheduled by using backfilling algorithm. The L3 is primitive in nature which is executed in the time slot 12.00–12.20 PM by a duration of 20 min and 30 min in the time slot 1.00–1.30 PM. In backfilling a best effort lease should be selected from the best effort queue which can provide idle resource in the free slots where a deadline sensitive lease can be scheduled. Here the L3 provides idle resource to schedule the L4 which is a deadline sensitive lease. The selection of best effort lease from the best effort queue should be performed by a decision maker. In our proposed work, we use AHP as a decision maker in the backfilling algorithm. The decision maker is used if there is similar type of best effort lease like the L3. This is discussed in the next section.

4. Proposed work

4.1. Problem statement

The backfilling algorithm is discussed in Section 3. The backfilling schedules the deadline sensitive lease by selecting a best effort lease from best effort queue. The selected lease should provide idle resources. These idle resources are allocated to the newly arrived deadline sensitive lease to schedule. The selection of the lease from the best effort queue should be carried by a decision maker when there is more than one conjugative and similar type of best effort leases in best effort queue. The problem formulated on the following conditions to find a similar task such as: (1) The leases must be conjugative in sequence (L_i, L_{i+1},. .. L_{i+n}). (2) The start time (ST) of leases should be same. (3) The required number of nodes (Nodes) also must be same. (4) The deadline (DT) of these should be same and (5) The execution time (E_i) of lease L_i must be less than the execution time (E_{i+1}) of L_{i+1}.

The Table 2 shows the different leases information, where the lease L3 and L4 are similar type of leases by considering the above conditions. The challenge for the backfilling algorithm is to select whether lease L3 or L4 from the best effort queue to schedule the lease L5.

4.2. Analytic hierarchy process (AHP)

In our work, we have used AHP as a decision maker to handle the challenge discussed in the Section 4.1. It is flexible, simple and powerful tool for finding the rank among the criterions. It is also used to optimize human resource allocation problem (Saaty et al., 2007). In AHP, there is no need to build a complex expert system. The AHP allows to model a complex problem in a hierarchical structure showing the relationships of the goal, objectives (criteria), sub-objectives, and alternatives (Saaty, 2008). It uses relative importance and pairwise comparison to get a score value for each option based on the criterion (Saaty, 1990).

The AHP uses a hierarchical model to solve the complex problem. The hierarchical model is constructed by the goal, objective and alternatives. The relative values of the alternatives or criterions are set by Saaty Rating Scale (Saaty, 1990), which is shown in the Table 3. The pairwise comparison is evaluated and represented by a matrix, which is called pairwise comparison matrix. This matrix specifies the relative importance the criterions. Then, these values are computed to find the ranks among the criterions. According to these rank values the decisions are taken.

4.3. Proposed approach (backfilling algorithm using AHP)

In this section, we proposed the mechanism to improve the backfilling algorithm for scheduling, using the AHP. The AHP is used as a decision maker in the backfilling algorithm to schedule the deadline sensitive leases. As we discussed in the Section 4.1, the backfilling algorithm needs to select a leases from the best effort queue which will provide free resources to schedule the newly arrived deadline sensitive lease. The challenge occurred, when there are similar types of leases in the best effort queue. We have used the AHP in the backfilling algorithm to overcome the challenge. The AHP is applied to these similar types of leases to select the exact lease which will provide free resources.

The backfilling algorithm is based upon the slack value of the lease. The Slack value is introduced to verify, where the lease can be scheduled or not using the backfilling algorithm. This value of the lease specifies, whether the lease is considered as a deadline sensitive lease or an advance reservation lease (Nathani et al., 2012). It decides the preemptability of the lease. The deadline, start time and duration of the lease are used as the parameters to calculate the slack value of a lease.

$$\text{Slack Value} = \frac{\text{deadline} - \text{start time}}{\text{duration}}$$

If the calculated slack value of a lease is greater than the system administrator decided lower bound slack value, then the lease is considered as deadline sensitive lease. In our proposed work the assigned lower bound slack value is 1.1 (Nathani et al., 2012). In the Table 2, all the leases have the slack value greater than 1.1. So all the leases should be scheduled using backfilling algorithm. But in the backfilling a task

Table 2 Lease Information.

Lease no.	Nodes	Submit time (AM)	Start time (PM)	Duration	Deadline (PM)
1	2	11.10	12.00	20	12.30
2	3	11.20	12.00	40	01.00
3	2	11.30	12.00	30	01.50
4	2	11.32	12.00	40	01.50
5	4	11.40	01.00	20	01.50

Table 3 Saaty Rating Scale.

Intensity	Importance
1	Equal importance
2	Weak or slight
3	Moderate importance
4	Moderate plus
5	Strong importance
6	Strong plus
7	Very strong
8	Very, very strong
9	Extreme importance

should be selected from the best effort queue which is pre-emptive. For which we introduced the AHP as a decision maker to select this lease.

4.3.1. Description of proposed scheme

In this section, we discussed the use of AHP to overcome the challenge which is described in the Section 4.1. In this proposed work, we considered the deadline, duration and start time as important parameters for scheduling the lease. So, the deadline, duration and start time of the deadline sensitive lease are considered as criterions/alternatives for the AHP in the proposed scheme. Similar type of leases are used as alternatives of the AHP. The selection of lease is also set as the goal of the AHP. The Decision Hierarchy Tree (DHT) of the proposed scheme of the illustration is constructed and shown in the Fig. 4.

The correlation values among the deadline, duration and start time are set, using Saaty Scale. These values are represented in a square matrix which specifies the relation between them. The square matrix is called correlation matrix. The correlation matrix is represented as below.

$$\begin{array}{c|cccc} & \mathrm{A} & \mathrm{B} & \mathrm{C} & \ldots\ \mathrm{N} \\ \hline \mathrm{A} & a_{11} & a_{12} & a_{13} & \ldots\ a_{1n} \\ \mathrm{B} & a_{21} & a_{22} & a_{23} & \ldots\ a_{2n} \\ . & . & . & . & \\ . & . & . & . & \\ . & . & . & . & \\ \mathrm{N} & a_{n1} & a_{n2} & a_{n3} & \ldots\ a_{nn} \end{array}$$

In the above matrix A, B, C,. .. N represents the criterions. The value of a_{11}, a_{12},. .. a_{nn} are the scale values which represent the correlation in between any two criterions, where $0 \geqslant a_{ij} \leqslant 9$.

Moreover, the pairwise comparison method is used to find the other correlation values among the deadline, duration and start time of the leases. The matrix A is a $n \times n$ real matrix, where n is the number of parameters in the decision hierarchy. The each entry a_{ij} of the matrix A represents the importance of the ith parameter relative to the jth parameter as:

(1) If $a_{ij} > 1$, then ith parameter is more important than jth parameter in matrix A.
(2) If $a_{ij} < 1$, then ith parameter is less important than the jth parameter.
(3) If a_{ij} is 1 then two parameters have the same importance. The entries a_{ij} and a_{ji} must satisfy the following constraint as:

$$a_{ij.} \cdot a_{ji} = 1$$

The computed pairwise matrix A with related weight matrix W for the deadline, duration and start time can be represented as:

$$\mathrm{A}_{n\times n} = \begin{pmatrix} \frac{a_{11}}{a_{11}} & \frac{a_{11}}{a_{12}} & \frac{a_{11}}{a_{13}} & \cdots & \frac{a_{11}}{a_{1n}} \\ \frac{a_{21}}{a_{11}} & \frac{a_{21}}{a_{12}} & \frac{a_{21}}{a_{13}} & \cdots & \frac{a_{21}}{a_{1n}} \\ & . & . & . & \\ & . & . & . & \\ & . & . & . & \\ \frac{a_{n1}}{a_{11}} & \frac{a_{n1}}{a_{12}} & \frac{a_{n1}}{a_{13}} & \cdots & \frac{a_{n1}}{a_{1n}} \end{pmatrix}$$

$$W_{n\times 1} = \begin{pmatrix} W_1 \\ W_2 \\ \cdots \\ \cdots \\ \cdots \\ W_n \end{pmatrix}$$

The eigenvector is computed for finding the relative weights (Saaty, 2003). Let λ is the right eigenvector of A and W if $AW = \lambda W$. Some inconsistencies may arise during performing the pairwise comparisons. To find the consistency among the parameters, we computed consistency index (CI). Let matrix A and B are involved in the pairwise comparison where the technique involved for finding inconsistency is only for matrix A. The matrix B can be obtained by replacing A with B. The Consistency Index (CI) is calculated by computing the eigenvector λ as the average of the elements of the vector whose jth element is the ratio of the jth element of the vector $A*w$ to the corresponding element of the vector w. Then the CI can be calculated as:

$$CI = \frac{\lambda_{\max} - n}{n - 1}\lambda_{\max}$$

The $CI = 0$ or $\lambda_{\max} = \mathrm{n}$ for all perfectly consistent decision maker. But the small value of inconsistency may be ignored if,

$$\frac{CI}{RI} < 0.1$$

where the RI *is* the Random Index, which is the consistency index when the entries of A are completely random.

Moreover, the matrix A is said to be perfectly consistent if and only if, the induced matrix with size n is $D = AA - nA$. The pair wise comparison matrix A is nearly consistent if, the induced matrix D is close to a zero matrix. Similarly, a pairwise matrix is inconsistent if the induced matrix D consists of some inconsistent elements which are deviating far away from zero (Ergu et al., 2011a,b). The consistency between the elements of the matrix specifies the correlation between them.

4.3.2. Proposed algorithm

The motivation of the proposed algorithm is to improve scheduling by allocating more number of leases using AHP in backfilling algorithm. The AHP is used as a decision support system when conflict arises among the similar types of leases in the backfilling algorithm to schedule the leases. AHP can be used as a time dependent decision maker (Saaty, 2007).

The flow chart of the proposed mechanism is shown in the Fig. 3. As per the principle of backfilling algorithm the 1st lease from the best effort queue is executed, then it finds the next lease (L_i) which will be fitted with the free slots at the time slot T_i. Before selecting the lease L_i to schedule in time T_i, the proposed mechanism test the next lease (T_{i+1}) whether it is similar to the lease (L_i) or not. If $L_i \,!= L_{i+1}$, then L_i will be scheduled according to the principle of backfilling algorithm else similar tasks rank will be evaluated by AHP. According to the rank, the best-effort queue is modified and leases are scheduled further. This process is repeated till the best effort queue is empty. The task should be scheduled within a deadline otherwise deleted from the queue.

4.3.3. Illustration

In this section, the proposed algorithm is discussed with example. To describe the proposed algorithm, we used the data shown in the Table 2. In the Table 2, lease 4 is nearly similar to the lease 3. Whenever, we schedule these leases a conflict arises in the backfilling algorithm to schedule lease 3 whether, before lease 4 or after.

As, we discussed the slack value in the Section 4.3, all leases in the Table 2 are deadline sensitive leases. These leases should be scheduled by using backfilling algorithm, but it fails due to conflict in between lease 3 and lease 4. To solve this, we used AHP as a decision maker in backfilling algorithm.

The AHP provides decision by constructing the decision hierarchy tree if the problem has more than one alternative. The decision is made according to the criterions which are finally used to evaluate the priority among the alternatives (Saaty, 1990). The decision hierarchy of the problem is constructed and shown in the Fig. 3. The selection of *lease* is set as the goal, the *deadline*, *duration* and the *start time* of the lease are set as criterions and at last the lease 3 (*L3*) and the lease 4 (*L4*) are set as alternatives in the AHP.

In our illustration lease 3 and lease 4 are similar types of leases. A decision support is required to select any one among these two leases for which *lease* is set as the goal in the decision hierarchy as shown in Fig. 3. Similarly, the decisions are made according to the criterions in AHP. In this proposed work we considered the *deadline*, duration (execution time) and *start time* are the constraint parameters of a deadline sensitive lease. The scheduling of deadline sensitive leases completely depends upon these parameters. To evaluate a correct decision we considered *deadline*, *duration* and *start time* of the lease in the decision hierarchy. The problem formulated for the illustration is selecting the correct lease among leases 3 and 4 which will provide idle resource. So, there are only two alternatives (lease 3 and lease 4).The schedule of other leases depend upon the scheduling sequence of lease 3 and lease 4. So lease 3 (*L3*) and lease 4 (*L4*) are alternatives in the decision hierarchy.

The vector of criterions is computed by using Saaty Rating Scale as shown in the Table 3. The co-relation values among the criterions are shown in a matrix below:

$$\begin{array}{cc} & \begin{array}{ccc} \text{Deadline} & \text{Duration} & \text{Start Time} \end{array} \\ \begin{array}{l} \text{Deadline} \\ \text{Duration} \\ \text{Start Time} \end{array} & \begin{pmatrix} 1 & 5 & 3 \\ & 1 & 3 \\ & & 1 \end{pmatrix} \end{array}$$

For computing the above criterion matrix we used the following properties of a deadline sensitive lease to find the scale value.

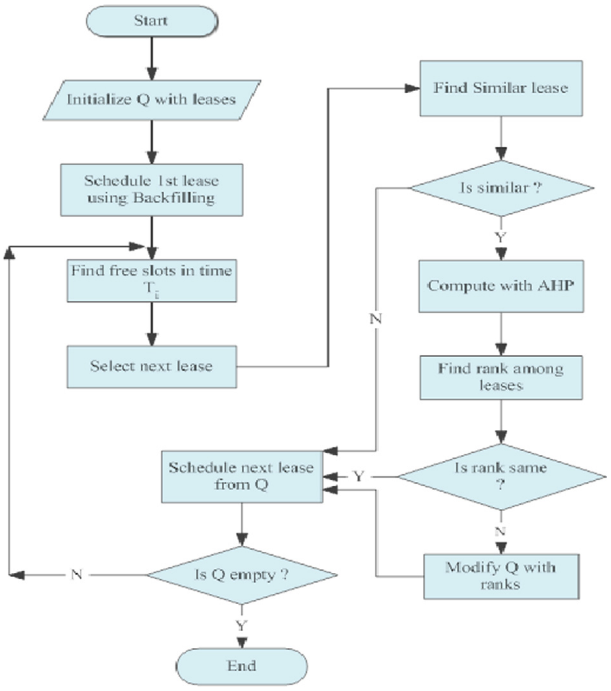

Figure 3 Flow chart of the proposed mechanism.

(1) *Deadline* is five times *Strong importance* than *Duration*. In the case of a deadline sensitive lease the lease should be completed with the predetermined time period. To meet the deadline, a lease must have to execute the complete *duration time*. If a lease L does not meet its deadline D, can be express as:

   Duration > deadline − Starttime

(2) *Deadline* is three times *Moderate important* than *Start Time*. In a deadline based lease a deadline can related to start time and duration as:

   Deadline >= Starttime + Duration + β

   ($\beta >= 0$)

   where, β is the additional time or time gap between the *finish time* and *Deadline*.

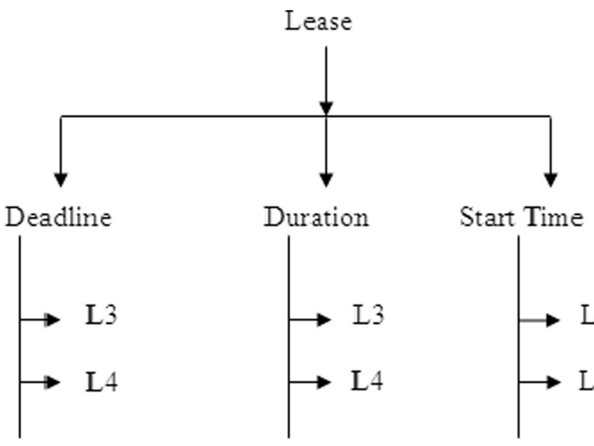

Figure 4 Decision Hierarchy of the problem.

(3) Similarly, we also considered *Duration* is three times *Moderate important* than *Start Time*. A lease can be scheduled at the exact time of start time, if the resources are idle. The pairwise comparison is found out of the above criterion matrix. It is computed as, if a row vector $r_j = (a_{i1}, a_{i2}, ..., a_{in})$ represents the *i*th row of the original pairwise comparison matrix A then the *j*th column of the same matrix can be represented as a column vector,

$$c_j^T = (a_{1j}, a_{2j}, ..., a_{nj})^T$$

where c_j^T is the transpose vector of column vector c_j. So the computed matrix is:

$$\begin{array}{l|ccc} & \text{Deadline} & \text{Duration} & \text{Start Time} \\ \hline \text{Deadline} & 1 & 5 & 3 \\ \text{Duration} & 1/5 & 1 & 3 \\ \text{Start Time} & 1/3 & 1/3 & 1 \end{array}$$

The inconsistency of the pairwise comparison matrix is discussed in the Section 4.3.1. To make the matrix consistent, we computed to find the eigenvector. The dot product of the vector r_i and c_j^T in n dimension is evaluated. The dot product b of the two vectors can be computed as,

$$\begin{aligned} b &= r_j.c_j^T \\ &= (a_{i1}, a_{i2}, \ldots, a_{in}).(a_{1j}, a_{2j}, \ldots, a_{nj}) \\ &= (a_{i1}a_{1j}, a_{i2}a_{2j}, \ldots, a_{in}a_{nj}) \\ &= (a_{i1}a_{1j}, a_{i2}a_{2j}, \ldots, a_{in}a_{nj}) \end{aligned}$$

If the matrix A is created, we compute the normalized matrix A_{norm} of matrix A as,

$$\overline{a_{ij}} = \sum_{j=1}^{m} a_{ij}$$

And the weighted value among the criterions w can be evaluated as,

$$w_i = \frac{\overline{a_{ij}}}{\sum_{l-1}^{m} a_{il}}$$

So the eigenvector E_1 can be found for the problem is

$$E_1 = \begin{bmatrix} 0.6722 \\ 0.2112 \\ 0.1165 \end{bmatrix}$$

[We considered up to 4th place after decimal point throughout in the proposed work.]

The next computed eigenvector E_2 for the matrix A is

$$E_2 = \begin{bmatrix} 0.6484 \\ 0.2239 \\ 0.1275 \end{bmatrix}$$

The consistency ratio can be found out by performing $E_1 - E_2$

$$CI = \begin{bmatrix} 0.0238 \\ -0.0127 \\ -0.011 \end{bmatrix}$$

The consistency ratio among the criterions is very close. So, there is no need to compute further and the last computed eigenvector E_2 is the rank matrix.

Similarly the eigenvectors for each criterion are computed. In the Section 4.3.1, we considered three criterions the deadline, the duration and the start time. Similarly, we have computed rank matrix E_2, the three criterion matrix are evaluated in same passion.

$$\text{For Deadline criterion} : E_{dl} = \begin{bmatrix} 0.2499 \\ 0.7500 \end{bmatrix}$$

$$E_{du} = \begin{bmatrix} 0.2499 \\ 0.7500 \end{bmatrix} \text{ for Duration.}$$

$$E_{st} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \text{ for Start time.}$$

To make the correct decision first the matrix E of $m \times n$ is computed from E_{dl}, E_{du}, and E_{st} eigenvectors where m is alternatives and n is objectives.

$$E = \begin{bmatrix} 0.2499 & 0.2499 & 0.5 \\ 0.7500 & 0.7500 & 0.5 \end{bmatrix}$$

Then the decision matrix D can be computed as,

$$D = E * E_2$$

$$= \begin{bmatrix} 0.2816 \\ 0.7195 \end{bmatrix}$$

The above matrix D shows lease 4 (L4) has more ranking value than lease 3 (L3). So, the lease 4 should be scheduled before lease 3. The scheduling results are shown in the Figs. 11 and 12.

5. Performance and result analysis

5.1. Performance

The performance of the proposed algorithm for scheduling of deadline sensitive leases is evaluated by comparing with the existing backfilling algorithm in which AHP is not used. The proposed work provides better scheduling of leases as compared to the existing backfilling algorithm in which, the similar leases conflict is not resolved.

The performance of the proposed mechanism is evaluated in terms of number of leases scheduled, number of leases rejected, total allocated time slots (AT) in minutes and wastage time slots (WT) in minutes. The total time slot (TS) is required to schedule n leases in a set may be calculated in minutes as:

$$TS = \max(N) * (\max(DT) - \min(ST))$$

$AT = \sum_{i=1}^{n}(N_i * ET_i)$ if L_i meets its deadline.

And the total wastage time slot (WT):

$$WT = TS - AT$$

We have considered three cases of experiments to evaluate the correctness, performance and efficiency of the proposed mechanism. The experiments are: (1) Number of leases is similar and deadlines are adjusted. (2) Deadlines are not adjusted.

(3) No similar leases. The experiments consist of a number of random numbers of leases with their required parameters shown in Tables 4, 6 and 7.

Experiment 1: Table 4 consists of different number of leases with different node requirements and also having a number of similar leases as we discussed in Subsection 4.1. Here, we adjusted the deadline time of the leases in minimum, by which all the leases are scheduled and able to meet their deadline. So, no leases are rejected in the proposed mechanism. But even if we adjusted the deadline some of the leases are rejected in the existing backfilling algorithm. We also observed that, whenever a similarity among the leases occurred, it leads to miss the deadline for the next immediate lease in the backfilling algorithm.

The percentages of allocated slots and wastage of slots of Table 4 are shown in the Table 8, which shows more resource utilization by allocating more number of leases. It also shows the wastage of time slots, which are reduced in the proposed method for each set of leases as compared to the backfilling algorithm.

Experiment 2: In this experiment, we tested the proposed work with different sets of leases. Here we have not adjusted the deadline of leases and also not resolved all conflicts that occurred due to similarity. Table 5 shows more leases are scheduled for the proposed method and less number of leases is rejected as compared to the backfilling algorithm. This specifies that more resources are utilized in the proposed method. The performance results of the leases scheduled and rejected are shown in the Fig. 9.

Experiment 3: This experiment is carried out to find the correctness of the proposed work. Though the backfilling is a standard algorithm, we compared the proposed algorithm with it. Here, we have been tested by a set of leases. There is no similarity among the leases which are shown in Table 7. The number of leases scheduled and rejected is same in the backfilling algorithm and proposed method.

5.2. Result analysis

The proposed work is implemented in MATLAB R2010a. The proposed work aims to schedule more number of leases by resolving the conflicts that arise due to similarity. The sets of leases are taken randomly along with similar leases. We also considered different number of virtual machines as shown in Tables 4, 6 and 7.

Fig. 5 shows the number of leases scheduled in experiment 1, where more than two leases are similar. All the leases are scheduled and the similarity conflicts are resolved in the proposed mechanism whereas some leases are rejected in the existing backfilling algorithm. Similarly more time slots are allocated to the leases in the proposed mechanism as compared to the backfilling algorithm which is shown in Fig. 6.

Though the proposed mechanism is able to schedule more number of leases, it utilizes more time slots. So the total

Table 4 (Experiment 1) Number of leases scheduled and rejected where deadline is adjusted.

Experiment number	No. of VM	No. of leases	No. of similar leases	No. of similar groups	Backfilling		Proposed method	
					No. of leases scheduled	No. of leases rejected	No. of leases scheduled	No. of leases rejected
1	4	5	2	1	4	1	5	0
2	4	7	4	1	6	1	7	0
3	4	10	4	2	8	2	10	0
4	4	20	8	4	16	4	20	0
5	6	10	4	2	8	2	10	0
6	6	36	21	7	29	7	36	0
7	6	50	28	6	44	6	50	0
8	8	15	6	3	12	3	15	0
9	8	30	16	4	26	4	30	0
10	8	50	20	7	43	7	50	0

Table 5 Number of allocated and wastage slots of experiment 1.

Experiment number	No. of VM	No. of leases	Total time slots in minutes (TS)	Backfilling		Proposed method	
				Allocated slots in minutes (AT)	Wastage slots in minutes (WT)	Allocated slots in minutes (AT)	Wastage slots in minutes (WT)
1	4	5	550	300	250	380	170
2	4	7	580	460	120	540	40
3	4	10	880	600	280	760	120
4	4	20	1760	1200	560	1520	240
5	6	10	1380	860	520	1100	280
6	6	36	4080	3070	1010	3910	170
7	6	50	6380	4790	1590	5870	510
8	8	15	2640	1680	960	2160	480
9	8	30	4100	2740	1360	3700	400
10	8	50	7520	4660	2840	6060	1460

Table 6 (Experiment 2) Number of leases scheduled and rejected where deadline is not adjusted (all similarity conflicts are not resolved).

Experiment number	No. of VM	No. of leases	No. of similar leases	No. of similar groups	Backfilling		Proposed method	
					No. of leases scheduled	No. of leases rejected	No. of leases scheduled	No. of leases rejected
1	4	20	8	4	8	12	10	10
2	4	25	10	5	10	15	12	13
3	4	30	12	6	11	19	13	17
4	4	40	14	8	17	23	19	21
5	5	25	10	5	10	15	12	13
6	5	30	15	6	12	18	15	15
7	5	55	27	8	21	34	29	26
8	7	12	4	2	8	4	11	1
9	7	30	14	4	13	17	17	13
10	7	44	22	6	23	21	27	17

Table 7 (Experiment 3) Number of leases scheduled and rejected (No similarity among the leases).

Experiment number	No. of VM	No. of leases	No. of similar leases	No. of similar groups	Backfilling		Proposed Method	
					No. of leases scheduled	No. of leases rejected	No. of leases scheduled	No. of leases rejected
1	4	10	0	0	7	3	7	3
2	4	20	0	0	8	12	8	12
3	5	10	0	0	6	4	6	4
4	6	24	0	0	11	13	11	13
5	7	40	0	0	17	23	17	23

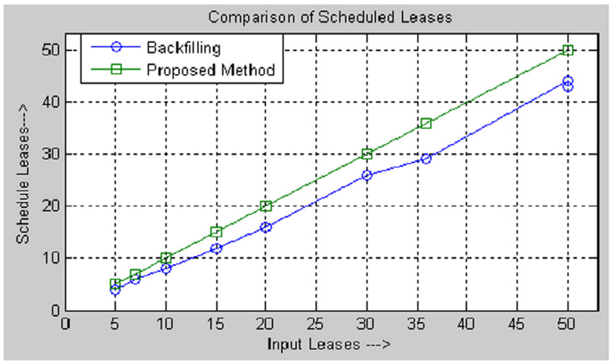

Figure 5 Number of leases scheduled in Experiment 1.

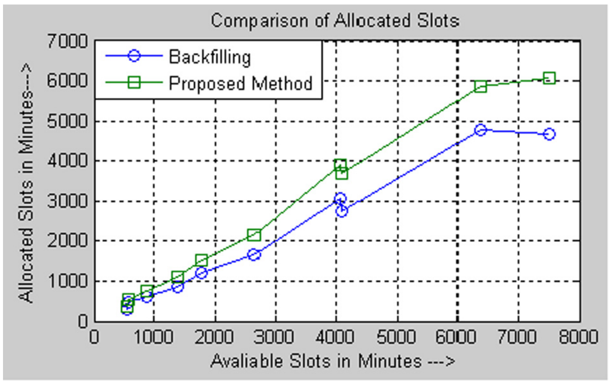

Figure 6 Number of time slots allocated in Experiment 1.

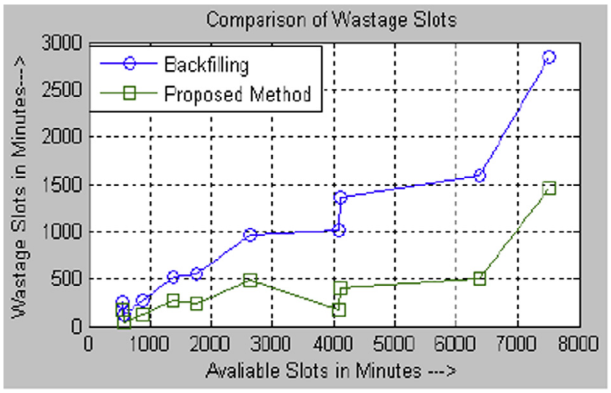

Figure 7 Number of wastage time slots in Experiment 1.

wastage of time slot in a physical machine is less than the total wastage of time slot in the backfilling algorithm which is shown in Fig 7. In Table 8, the percentage of resource utilization in the proposed work for experiment 1 is quite more than the backfilling algorithm. This specifies that the proposed mechanism provides better resource utilization. The comparison graph of resource utilization is shown in Fig. 8.

Fig 9 shows the leases scheduled in experiment 2. In the experiment 2 we have not resolved all the similarity conflicts. Some of the conflicts are resolved and some of are not. If we will change the deadline of the leases for minimum amount of time, then the out will be like experiment 1. Though the deadlines are not adjusted, there is less difference of scheduled leases in between the proposed method and the backfilling algorithm. The difference occurred due to resolving some of the conflicts.

Table 8 Comparison of Resource Utilization in Experiment 1.

Experiment number	Total time slots in minutes (TS)	Utilization of slots in%		Wastage of slots in%	
		Backfilling algorithm	Proposed method	Backfilling algorithm	Proposed method
1	550	54.54%	69.09%	45.54%	30.9%
2	580	79.31%	93.10%	20.68%	6.89%
3	880	68.18%	86.36%	31.81%	13.63%
4	1760	68.18%	86.36%	31.81%	13.63%
5	1380	62.31%	79.71%	37.68%	20.28%
6	4080	75.24%	95.83%	24.75%	4.17%
7	6380	75.07%	92%	24.92%	8%
8	2640	63.63%	81.81%	36.36%	18.18%
9	4100	66.82%	90.24%	33.17%	9.76%
10	7520	61.96%	80.58%	37.76%	19.41%

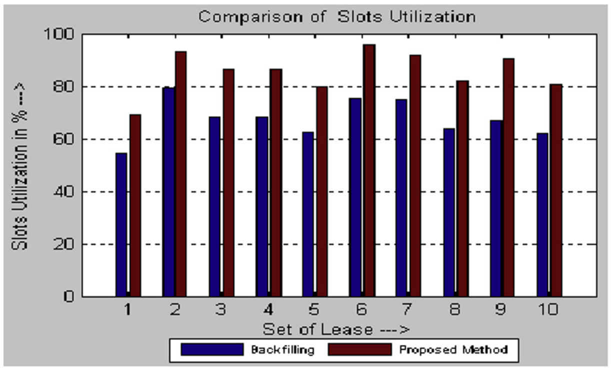

Figure 8 Comparison of Resource Utilization in Experiment 1.

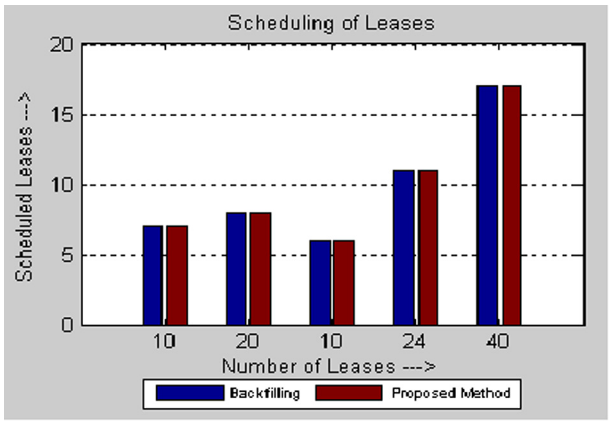

Figure 10 Comparison of leases scheduled in Experiment 3.

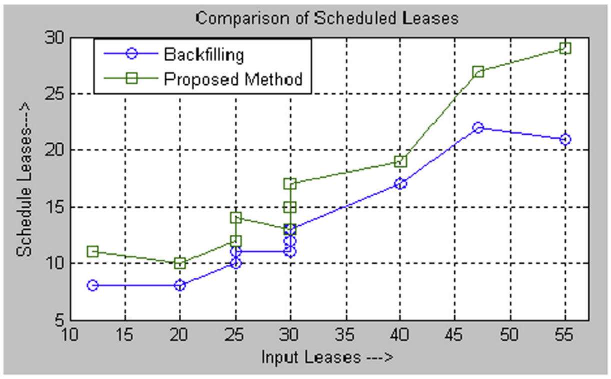

Figure 9 Scheduling of leases in Experiment 2.

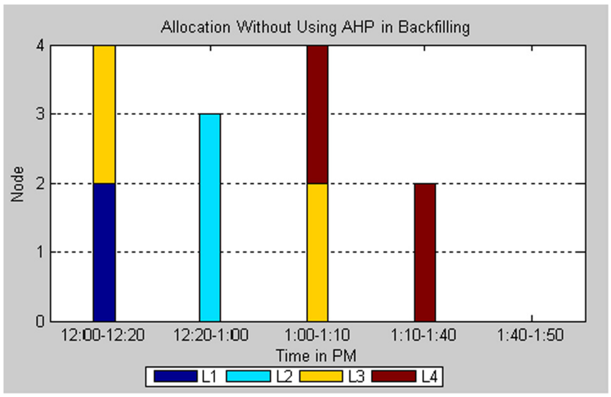

Figure 11 Scheduling of leases using the existing backfilling algorithm (without AHP).

In experiment 3, we considered a number of leases of sets without similarity among the leases as shown in the Table 7. Since there is no similarity the proposed algorithm behaves like backfilling algorithm. The number of leases scheduled in the proposed mechanism is equal to that of the backfilling algorithm.

6. Conclusion and future work

As a decision maker AHP is used in different areas. It takes the correct decision using the scale values. The mechanism is simple and robust to implement. In this proposed work, we have implemented AHP within the backfilling algorithm as a decision maker. The backfilling algorithm schedules the leases in first come first served basis if more than two leases are similar. It reduces the performance of backfilling as shown in Figs. 5, 6 and 9. The proposed work resolves conflicts using AHP which evaluate ranks among the similar leases and schedules more number of leases as shown in Figs. 5 and 9. The wastage of slots is also minimized as compared to the backfilling algorithm as shown in the Fig. 7. The proposed algorithm

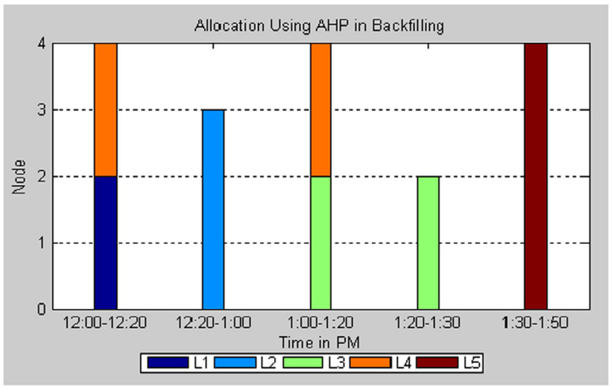

Figure 12 Scheduling of leases using AHP and backfilling algorithm.

improves the performance of the backfilling algorithm if there are similar leases. Otherwise it is same as the backfilling algorithm shown in Fig. 10.

Moreover, the backfilling algorithm can be implemented and modified using AHP in Haizea for OpenNebula in future. Except AHP, other decision makers can also be used in the backfilling algorithm and the scheduling performance can be studied. We are also working to propose a new algorithm to overcome the disadvantages of the backfilling algorithm in the near future.

Acknowledgments

The authors would like to thank Dr. P. K. Pattnaik, School of Computer Engineeirng, KIIT University and Mrs. S. Parida, C.V. Raman College of Engineering for their suggestions and views given during this work. I would also like to thank the Management of C.V. Raman College of Engineering for providing full support to me.

References

Abrishami, S., Naghibzadeh, M., Epema, D., 2013. Deadline constrained workflow scheduling algorithms for IaaS clouds. Future Gener. Comput. Syst. 29, 158–169.

Kyu, Byun, Kee, Yang-Suk, Kim, Jin-Soo, Maeng, Seungryoul, 2011. Cost optimized provisioning of elastic resources for application workflows. Future Gener. Comput. Syst. 27 (8), 1011–1026. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0167739X11000744>.

Calheiros, R.N., Buyya, R., 2014. Meeting deadlines of scientific workflows in public clouds with tasks replication. IEEE Trans. Parallel Distrib. Syst. 25 (7), 1787–1796.

Dong, F., Akl, S.G., 2006. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems, pp. 1–55.

Ergu, Daji, Gang, Kou, Yi, Peng, Yong, Shi, 2011a. A simple method to improve the consistency ratio of the pairwise comparison matrix in ANP. Eur. J. Oper. Res. 213 (1), 246–259.

Ergu, Daji, Kou, Gang, Peng, Yi Shi, Yong, Shi, 2011b. The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment. The Journal of Supercomputing, 835–848.

Feitelson, D.G., 1998. Utilization and Predictability in Scheduling the IBM SP2 with Back lling. In: 12th Intl. Parallel Processing Symp. IEEE, pp. 542–546.

Haizea.cs.uchicago.edu, 2016. Haizea – An Open Source VM-Based Lease Manager". N.p., 2016. Web. 15 Jan. 2016.

Kalra, M., Singh, S., 2015. A review of metaheuristic scheduling techniques in cloud computing. Egypt. Inf. J. 16 (3), 275–295. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S1110866515000353>.

Li, X., Cai, Z., 2015. Elastic Resource Provisioning for Cloud Workflow Applications. IEEE Trans. Autom. Sci. Eng., 1–16 Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7352380>.

Lifka, D.A., 1995. The ANL/IBM SP scheduling system. In: IPPS'95: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing. Springer-Verlag, London, UK, pp. 295–303.

Liu, Jinzhao, Zhang, Yaoxue, Zhou, Yuezhi, Zhang, Di, Liu, Hao, 2014. Aggressive Resource Provisioning for Ensuring QoS in Virtualized Environments. IEEE Transactions on Cloud Computing, PP(99), pp.1–1. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6888495>.

López, J.M., García, M., Díaz, J.L., García, D.F., 2003. Utilization bounds for multiprocessor rate-monotonic scheduling. Real Time Syst. 24 (1), 5–28.

Manimaran, G., Murthy, C.S.R., 1998. An efficient dynamic scheduling algorithm for multiprocessor\nreal-time systems. IEEE Trans. Parallel Distrib. Syst. 9 (3), 312–319.

Nathani, A., Chaudhary, S., Somani, G., 2012. Policy based resource allocation in IaaS cloud. Future Gener. Comput. Syst. 28 (1), 94–103. http://dx.doi.org/10.1016/j.future.2011.05.016.

Opennebula.org, 2016. Opennebula|Flexible Enterprise Cloud Made Simple. N.p., 2016. Web. 15 Jan. 2016.

Pandey, Suraj, Wu, Linlin, Guru, Siddeswara Mayura, Buyya, Rajkumar, 2010. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: Proceedings - International Conference on Advanced Information Networking and Applications, AINA, (July), pp. 400–407.

Rodriguez, M.A., Buyya, R., 2014. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. IEEE Trans. Cloud Comput. 2 (2), 222–235.

Saaty, T.L., 1990. How to make a decision: the analytic hierarchy process. Eur. J. Oper. Res. 48 (1), 9–26.

Saaty, T.L., 2003. Decision-making with the AHP: Why is the principal eigenvector necessary. Eur. J. Oper. Res. 145 (1), 85–91. Available at: <http://www.sciencedirect.com/science/article/pii/S0377221702002278> [accessed 16.01.16].

Saaty, T.L., 2007. Time dependent decision-making; dynamic priorities in the AHP/ANP: generalizing from points to functions and from real to complex variables. Math. Comput. Modell. 46 (7–8), 860–891. Available at: <http://www.sciencedirect.com/science/article/pii/S0895717707000982> [accessed 03.12.15].

Saaty, T.L., 2008. Decision making with the analytic hierarchy process. Int. J. Serv. Sci. 1 (1), 83–98.

Saaty, T.L., Peniwati, K., Shang, J.S., 2007. The analytic hierarchy process and human resource allocation: half the story. Math. Comput. Modell. 46 (7–8), 1041–1053. Available at: <http://www.sciencedirect.com/science/article/pii/S0895717707000842> [accessed 16.01.16].

Sotomayor, B., Keahey, K., 2007. Enabling cost-effective resource leases with virtual machines. In: Hot Topics Session, pp. 16–18. Available at: <https://phoenixforge.cs.uchicago.edu/svn/haizea/branches/TP2.0/html/pubs/Haizea_HPDC2007HotTopics.pdf>.

Sotomayor, B., 2009. The Haizea Manual. Science.

Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I., 2008. Capacity Leasing in Cloud Systems using the OpenNebula Engine. In: Most, pp. 1–5. Available at: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Capacity+Leasing+in+Cloud+Systems+using+the+OpenNebula+Engine#0>.

Sotomayor, Borja, Montero, Ruben Santiago, Llorente, Ignacio Martin, Foster, Ian, 2009a. Virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput. 13, 14–22.

Sotomayor, Borja, Montero, Ruben Santiago, Llorente, Ignacio Martin, Foster, Ian, 2009b. Resource Leasing and the Art of Suspending Virtual Machines. In: 2009 11th IEEE International Conference on High Performance Computing and Communications, pp. 59–68. Available at: < http://www.computer.org/portal/web/csdl/doi/10.1109/HPCC.2009.17 > .

Wu, Z., Ni, Z., Gu, L., Liu, X., 2010. A revised discrete particle swarm optimization for cloud workflow scheduling. In: Proc. IEEE Int. Conf. Comput. Intell. Security, pp. 184–188.