Azerbaijan Journal of High Performance Computing, Vol 2, Issue 1, 2019, pp.29-38
https://doi.org/10.32010/26166127.2019.2.1.29.38

*Correspondence:
Shamsollah Ghanbari,
Islamic Azad University,
Ashtian Branch, Ashtian,
Iran, myrshg@gmai.com

Priority-aware Job Scheduling Algorithm in Cloud Computing: A Multi-criteria Approach

Shamsollah Ghanbari
Islamic Azad University, Ashtian Branch, Ashtian, Iran, myrshg@gmai.com

Abstract

Job scheduling is one of the most problematic theoretical issues in the area of cloud computing. The existing scheduling methods attempt to consider only a few criteria of scheduling without covering other sufficient criteria. Since, cloud computing faces a large scale resource for allocating to a large number of jobs, due to optimizing the users' requirements; therefore, a suitable cloud-based job scheduling method must satisfy a wide range of criteria. Besides, in cloud computing, the jobs come with different priorities. Thus, in the cloud environment, a suitable job scheduling algorithm should be able to combine several priorities. This paper proposes a new multi-criteria priority-aware job scheduling algorithm in cloud computing. Experimental results indicate that the proposed method is able to consider different criteria for scheduling.

Keywords: cloud computing, multi-criteria, priority-aware job scheduling.

1. Introduction

Cloud computing is a large-scale distributed computing assumption that is motivated by economies of scope, in which a pool of abstracted, dynamically-scalable, simulated, storage, platforms, managed computing power, and services are given on demand to external consumers through the internet. Scheduling of jobs is one of the most problematic theoretical issues in the area of cloud computing. The main goal of job scheduling is to obtain high performance computing and the best system throughput [1]. The primary job scheduling algorithms including First Come First Serve(FCFS), Shortest Job First (SJF) [2], Shortest Remaining Job First (SRJF) [3, 4] and Round Robin [5] are not suitable methods for scheduling in the area of cloud computing. Because the basic job scheduling algorithms can fulfill a few metrics of scheduling. There is considerable literature to concern the scheduling of jobs in cloud computing. According to [6], the related algorithms can be categorized into two significant classes, including batch mode and online mode algorithms. In the batch mode algorithms, jobs are collected from the different users and queued when they arrive in the system. The scheduling will start after a fixed period. The other possible type of scheduling algorithms is called "Online mode heuristic" scheduling algorithm. In online mode algorithms, jobs are immediately scheduled when they arrive in the system. The online mode algorithms are more appropriate for the cloud environment because the cloud environment is a heterogeneous system.

Since, cloud computing considers a large scale resource to allocate a large number of jobs, in order to optimize the users' requirements. Therefore, a suitable cloud-based job scheduling method must fulfill different criteria [7, 8, 9, 10, 11, 12]. According to our knowledge, the following criteria can be considered for job scheduling algorithms in cloud-based environments.

- Performance criteria including makespan, throughput, latency, total completion time, total communication time, total cost, system utilization ant Qos [13].
- Job criteria including divisible load vs. nondivisible load [14], sequentially dependent vs. sequentially independent, priority based vs. non-priority based [15].
- Strategies criteria including batch mode vs online mode [6], predictable vs non-predictable [16], deterministic vs non deterministic [17].
- Resource criteria including heterogonous vs. homogenous [18], preemptive vs non-preemptive, dynamic vs static.
- Reliability and validity criteria including trust [19], fairness [20]
- Algorithm criteria including time complexity, space complexity [21].

In general, it is not possible to optimize all criteria for job scheduling using any algorithm. Typically, the criteria are prioritized, with most attention paid to the most important criterion. The existing algorithms can consider some particular criteria because of its nature. A multiple criteria algorithm can be much more useful for job scheduling in cloud computing. Accordingly, we need a multi-criteria approach for scheduling in the cloud environment. The two main types of multi-criteria decision-making model are multi-objective decision-making and multi-attribute decision-making. A multi-objective scheduling algorithm which is suitable for scheduling in big data, e.g., cloud computing has been proposed in [22].

In this paper, we have a multi-attribute decision-making approach to the problem. We propose a multi-level priority-based method of scheduling. The proposed method uses the theory of Analytical Hierarchy Process (AHP).

2. Analytical Hierarchy Process

In this section, we describe the Analytical Hierarchy Process (AHP). Generally, the AHP contains three major levels, including objective, attributes, and alternatives levels. Each level uses the comparison matrices to compare the priorities [23, 24]. Assume that $A = [a_{rt}]$ is a comparison matrix. Every entry in matrix A is positive. In this occasion, A is a square matrix $(A_{n\times n})$ There is only one vector of weights such as $u = (u_1, u_2, \dots, u_n)$ in association with any arbitrary comparison matrix including A. This vector also is named priority vector. The relationship between the elements of the comparison matrix (A) and its vector of weights (u) is demonstrated by Eq. (1).

$$a_{rt} = \begin{cases} \frac{u_r}{u_t} & r \neq t \\ 1 & r = t \end{cases} \quad (1)$$

According to [23, 24], the priority vector of matrix A can be computed with the following equation:

$$Av = \lambda_{max} \cdot v \quad (2)$$

where λ_{max} and v are the principal eigenvalue and the equivalent priority vector of A, respectively. If A is wholly consistent, then $\lambda_{max} = n$. There are diverse methods for clarifying the consistency of a comparison matrix [23, 24].

3. Preliminaries

We present a multi-criteria priority-aware method for scheduling the jobs in the area of cloud computing. The proposed method can assign jobs to resources based on the best priority. We consider that the users submit their request via sum cloudlets. A is mobility- enhanced small-scale cloud that is located at the edge of the Internet. The cloudlets put diverse priority label to the requests. The proposed method can combine the criteria of cloudlets for computing the best priority. Theorem 1 demonstrates how to mix the priorities of the jobs to assign the resources based on criteria of diverse cloudlets.

Theorem 1. Assume that $\pi^i = (\pi_1^i, \pi_2^i, \dots, \pi_j^i, \dots, \pi_m^i)$ is a priority vector. We suppose that $\pi^1, \pi^2, \dots, \pi^k$ are k priority vectors and $r_1, r_2, \dots, r_i, \dots, r_k$ are corresponding priority values, respectively. The best-approximated priority can be calculated as follows:

$$\pi = \begin{pmatrix} \pi_1^1 & \pi_1^2 & \cdots & \pi_1^k \\ \pi_2^1 & \pi_2^2 & \cdots & \pi_2^k \\ \vdots & \vdots & \vdots & \vdots \\ \pi_m^1 & \pi_m^2 & \cdots & \pi_m^k \end{pmatrix} \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_k \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{k} \pi_1^j r_j \\ \sum_{j=1}^{k} \pi_2^j r_j \\ \vdots \\ \sum_{j=1}^{k} \pi_m^j r_j \end{pmatrix} \tag{3}$$

Proof. Assume that $R = (r_1, r_2, \dots, r_i, \dots, r_k)$ be the equivalence eigenvector of a given comparison matrix denoted by $D_{k\times k}$. In fact, $D_{k\times k}$ is the comparison matrix of k criteria. We are going to compare m alternatives based on the k mentioned criteria. We also assume that $\pi^1, \pi^2, \dots, \pi^k$ are the corresponding eigenvectors of k comparison matrices. By using the theory of AHP and proposed theorem in [25], the best priority for the mentioned vectors can be computed.

The following theorem is a method for calculating the priority vector of a comparison matrix.

Theorem 2. Suppose A be a comparison matrix and *let*= $(1, 1, \dots, 1)^T$. Also supposed that u and $\lambda_{\max}$ are principal eigenvector and corresponding eigenvalue of A, respectively. The principal eigenvector of A can be computed as follows:

$$u = \lim_{\mu\to\infty} \frac{1}{\mu} \sum_{r=1}^{\mu} \frac{A^r e}{e^T A^r e} \tag{4}$$

$$\lambda_{max} = \lim_{\mu\to\infty} \frac{e^T A^{\mu+1} e}{e^T A^{\mu} e} \tag{5}$$

Proof. A proof for this theorem can be found in [23].

4. Proposed Method

A general framework for the proposed method has been depicted in Fig. 1. The proposed method consists of the following three phases:

1.Initializing. In the first phase, the jobs are sent by the users to the cloudlets. Each cloudlet puts a set of priority-stamp to the jobs for allocating the requested resources.

2. Priority-aware Resource Allocation. This phase is the main part of the proposed method. In this phase, a global scheduler combines the jobs based on the priority-stamps of the local cloudlets. This phase contains the following five steps:

• Step 1: (Initializing). Assume that $\Phi = \{J_1, J_2, \ldots, J_d\}$ is a collection of jobs that demands resources in a cloud-based environment. We also assume that $\Psi = \{R_1, R_2, \ldots, R_i\}$ is a collection of resources obtainable in cloud environment ($d \geq l$). Each

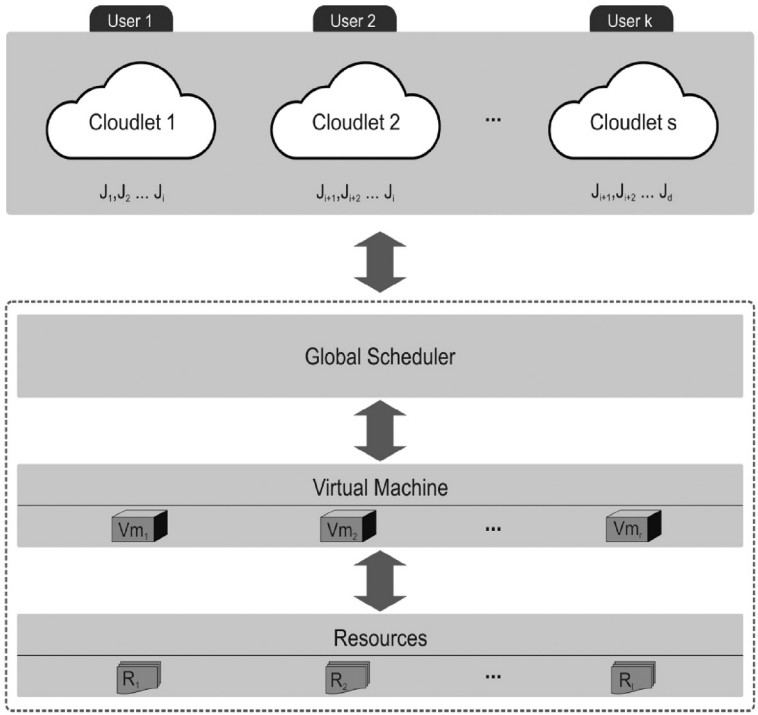


Fig. 1: The architecture of the proposed method.

job demands a subset of resources. At first, these queries are sent to the local cloudlets. Each local cloudlet puts a set of priority stamps based on different criteria on the requested jobs. Now assume that $\Upsilon_x(J_i, R_j)$ is the priority-stamp of the i^{th} job for assigning j^{th} reSource based on x^{th} criterion.

• Step 2: (Computing the Criteria Matrix). This step computes the priority-stamp of criteria for assigning the k^{th} resource. The criteria matrix is denoted by Υ^k. This matrix compares the priority-stamps of criteria based on the criterion of the global scheduler. The following equation can compute the Υ^k matrix:

$$\Upsilon^k[x, y] = \frac{P^x(\Upsilon^k)}{P^y(\Upsilon^k)} \quad (6)$$

where *x,y=1, 2, …, s* (s is the number of cloudlets). Moreover, $P^x(\Upsilon^k)$ and $P^y(\Upsilon^k)$ in Eq. 6, are the priority stamp of the x^{th} and y^{th} criteria based on the global scheduler.

• Step 3: (Making Comparison Matrices of the Jobs). This step, computes the

comparison matrices for the jobs to assign the resources based on the criteria of the cloudlets. The comparison matrices indicate the priorities of jobs based on different cloudlets.

The comparison matrices can be computed by the following equation:

$$\Lambda_{rt}^{kx} = \begin{cases} \frac{\Upsilon_\infty(J_r,R_k)}{\Upsilon_\infty(J_t,R_k)} & r \neq t \\ 1 & r = t \end{cases} \quad (7)$$

where Λ^{kx} is the comparison matrix of the $\mathrm{k}^{\mathrm{th}}(\mathrm{k} = 1,2,\dots,\mathrm{s})$ resource based on the criteria of $\mathrm{x}^{\mathrm{th}}(\mathrm{k} = 1,2,\dots,\mathrm{s})$ cloudlet.

• Step 4: (Computing the Priority Vector). This step computes priority vectors for the matrices of jobs. For this purpose, was used Algorithm 1.

Algorithm 1 Corresponding Principal Eigenvector ()

$1: \mathrm{D} \leftarrow \Upsilon^{\mathrm{k}}$

$2: \mu \leftarrow 1$

$3: \mathrm{e} \leftarrow (1,1,\dots,1)$

$4: \mathrm{h}_1 \leftarrow \frac{\mathrm{D}^{\mu}{}_{\mathrm{e}}}{\mathrm{e}^{\mathrm{T}}\mathrm{D}^{\mu}{}_{\mathrm{e}}}$

$5: \mathrm{h}_2 \leftarrow \mathrm{h}_1$

$6:$ while $(|\|\mathrm{h}_1\| - \|\mathrm{h}_2\|| \geq \epsilon)$ do

$7: \mu \leftarrow \mu + 1$

$8: \mathrm{h}_1 \leftarrow \mathrm{h}_2$

$9: \mathrm{h}_2 \leftarrow \mathrm{h} + \frac{\mathrm{D}^{\mu}{}_{\mathrm{e}}}{\mathrm{e}^{\mathrm{T}}\mathrm{D}^{\mu}{}_{\mathrm{e}}}$

$10:$ Return $\frac{\mathrm{h}}{\mu}$

Now, suppose that $\mathrm{q}^{\mathrm{k1}}, \mathrm{q}^{\mathrm{k2}}, \dots, \mathrm{q}^{\mathrm{kd}}$ are the priority vectors of $\Lambda^{k1}, \Lambda^{k2}, \dots, \Lambda^{kd}$. Therefore, we have d priority vectors have associated with the d comparison matrices for assigning the k^{th} resource.

• Step 5: (Computing the Priority of Jobs). By using Theorem 1, the priority of jobs for k^{th} resource denoted by δ^{k}, can be calculated by the following equation:

$$\delta^k = \Delta^k \times \Gamma^k \quad (8)$$

where Γ^k is the priority vector of Υ^k and Δ^k is defined as the following equation:

$$\Delta^k = [q^{k1} q^{k2} \dots q^{kd}] = \begin{pmatrix} q_1^1 & q_1^2 & \cdots & q_1^k \\ q_2^1 & q_2^2 & \cdots & q_2^k \\ \vdots & \vdots & \vdots & \vdots \\ q_d^1 & q_d^2 & \cdots & q_d^k \end{pmatrix} \quad (9)$$

3. Assigning the Final Priority Stamp. This step assigns the final priority stamp on the jobs. The jobs with the highest priority-stamps will be assigned to VMs for allocating the appropriate resources. Algorithm 2 indicates the details of the proposed method.

5. Performance Evaluation

In this section, were provided some experimental results. Now assume that $\Phi = \{J_1, J_2, J_3, J_4\}$ is a collection of jobs that demand resources in a cloud-based environment. We also suppose

Algorithm 2 Calculation Priority of Processors ()
Description:
1: $Let\ \Phi = \{j_1, j_2, \dots, j_d\}$
2: $Let\ \Psi = \{R_1, R_2, \dots, R_l\}$
3: $Compute\ the\ Priority\ Stamp\ based\ on\ Different\ Request\ Manager$
4: $Let\ T \leftarrow Time\ Slice$
5: $While\ T\ do$
6: $Make\ Criteria\ Matrix\ (\Upsilon^k)\ using\ Eq.6$
7: $Calculate\ \Gamma^k\ Priority\ Vector\ of\ \Upsilon^k\ using\ Algorithm\ 1$
8: $Make\ Resource\ Matrices\ (\Lambda^{kx}); (k = 1,2, \dots, l)\ and\ (x = 1,2, \dots, s)\ using\ Eq.7$
9: $For\ all\ Resource\ Matrices\ (\Lambda^{kx}); (k = 1,2, \dots, l)\ and\ (x = 1,2, \dots, s)$
$Calculate\ corresponding\ priority\ vector\ using\ Algorithm\ 1$
9: $Make\ \Delta^k\ using\ Eq.9$
10: $end\ while;$

that $\Psi = \{R_1, R_2, \dots, R_4\}$ is a set of resources available for the global scheduler. Also was considered four cloudlets with different criteria, including makes pan, throughput, time complexity, and trust, respectively.

Let.$\big(x, ps(x)\big)\epsilon\{(makespan, 0.8), (throughput, 0.3), (complexity, 0.2), (trust, 0.5)\}$ Thus the comparison matrix for the criteria of a good assignment to the k^{th} resource can be computed as follows:

$$\Upsilon^k = \begin{pmatrix} 1.000 & 0.375 & 0.250 & 0.625 \\ 2.667 & 1.000 & 0.667 & 1.667 \\ 4.000 & 1.500 & 1.000 & 2.500 \\ 1.600 & 0.600 & 0.400 & 1.000 \end{pmatrix}$$

By using Algorithm 1, the corresponding priority vector of Υ^k denoted by Γ^k can be computed by using the following equation:

$$\Gamma^k = \begin{pmatrix} 0.108 \\ 0.288 \\ 0.431 \\ 0.173 \end{pmatrix}$$

Moreover, the comparison matrices of jobs based on different criteria for assigning the k^{th} resource are shown in Tables 1-4.

Table 1: Comparison matrix based on makespan for assigning q^{k1} resource.

Jobs	Jobs				q^{k1}
	j_1	j_2	j_3	j_4	
j_1	1.000	0.666	0.333	1.333	0.160
j_2	1.500	1.000	0.500	2.000	0.240
j_3	3.000	2.000	1.000	4.000	0.480
j_4	0.750	0.500	0.250	1.000	0.120

By using Algorithm 1, the priority of jobs (denoted by δ^k) for assigning the k^{th} resource can be calculated by the following equation:

Table 2: Comparison matrix based on throughput for assigning q^{k2} resource.

Jobs	Jobs				q^{k2}
	j_1	j_2	j_3	j_4	
j_1	1.000	1.120	0.200	0.800	0.126
j_2	0.833	1.000	0.166	0.666	0.107
j_3	5.000	6.000	1.000	4.000	0.646
j_4	1.250	1.500	0.250	1.000	0.121

Table 3: Comparison matrix based on complexity for assigning q^{k3} resource.

Jobs	Jobs				q^{k3}
	j_1	j_2	j_3	j_4	
j_1	1.000	0.750	0.500	0.500	0.158
j_2	1.333	1.000	0.666	0.666	0.210
j_3	2.000	1.500	1.000	1.000	0.316
j_4	2.000	1.500	1.000	1.000	0.316

Table 4: Comparison matrix based on trust for assigning q^{k4} resource.

Jobs	Jobs				q^{k4}
	j_1	j_2	j_3	j_4	
j_1	1.000	2.000	4.000	8.000	0.533
j_2	0.500	1.000	2.000	4.000	0.267
j_3	0.250	0.500	1.000	2.000	0.133
j_4	0.125	0.250	0.500	1.000	0.067

$$\delta^k = \Delta^k \times \Gamma^k = \begin{pmatrix} 0.160 & 0.126 & 0.158 & 0.533 \\ 0.240 & 0.107 & 0.210 & 0.267 \\ 0.480 & 0.646 & 0.316 & 0.133 \\ 0.120 & 0.121 & 0.316 & 0.067 \end{pmatrix} \begin{pmatrix} 0.108 \\ 0.288 \\ 0.431 \\ 0.173 \end{pmatrix} = \begin{pmatrix} 0.214 \\ 0.193 \\ 0.397 \\ 0.196 \end{pmatrix}$$

The priority of jobs based on different criteria for assigning k^{th} resource has been shown in Fig. 2. The figure demonstrates how alternatives act on each criterion. The total priority (proposed method) of each alternative is where it intersects the axis on the right. The priority of each criterion is demonstrated by the rectangular box on that criterion's vertical fine, as read from the axis at the left.

Now is used cloudsim in order to simulate the results. The parameter settings are indicated in Table 5. We consider two different scenarios, including time shared and space shared. For each scenario, we compute the turn around time of the jobs. The results have been depicted in Fig. 3a and 3b. As the figures show, in both scenarios, the proposed method can decrease the time.

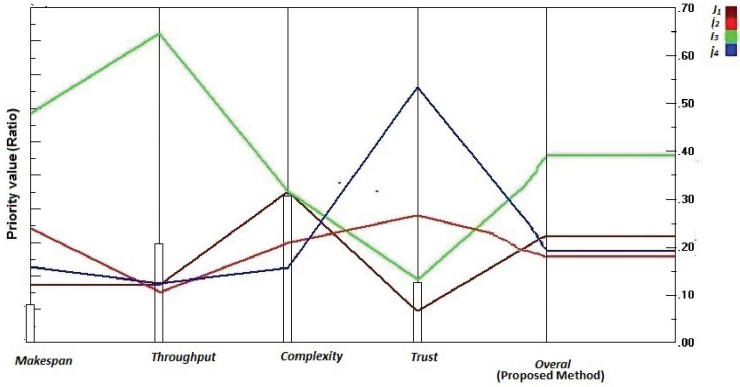
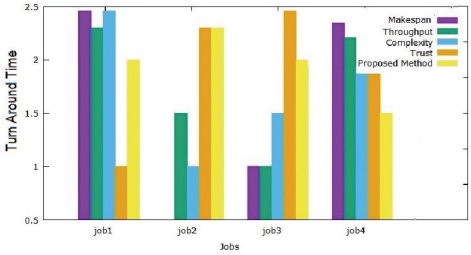


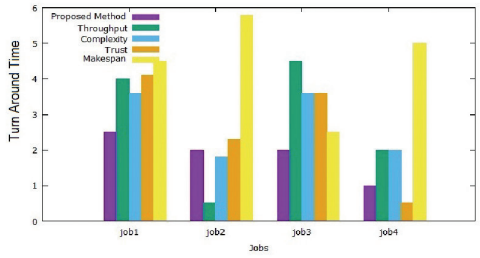
Fig. 2: The priority of jobs based on dierent criteria.

Table 5: Cloudlet specification

parameters	value
The Number of VMs	4
RAM	1024
MPs	1000
Bandwidth	1000



(a) Simulation results for the rst scenario (space shared)



(b) Simulation results for the second scenario (time shared)

6. Discussion

In this section, was computed the complexity of computation of the proposed method. In this situation, the complexity of the proposed method is equivalent to the number of computations that the cloudlets must execute. Thus, the complexity of computation in the proposed method is equal to the complexity of Algorithm 2. The following equation can compute it:

$$t_{prop} = c_1 \times t^{CPE}(s) + c_2 \times l \times t^{CPE}(d) \tag{10}$$

We also have:

$$t^{CPE}(s) = c_4 s^{2.81} \tag{11}$$

and

$$t^{CPE}(d) = c \times d^{2.81} \tag{12}$$

where c_1, c_2, c_3 and c_4 are four constant numbers. Moreover, t^{CPE} is the number of computations for computing Algorithm 1. The other notations, including s, d, and l are the number of criteria, jobs, and the resources respectively. Note that have been applied to Strassen's algorithm [26] for multiplication of two matrices in Eqs. (11) and (12).

7. Conclusion

This paper presented a multi-criteria priority-aware job scheduling algorithm, which is useful for the cloud-based environment. The proposed method can combine different criteria in order to find the best solution in cloud scheduling. We tested the proposed method in the CloudSim environment toolkit. The simulated results indicated that the proposed method could provide the best solution. We have also indicated that the proposed method has a reasonable complexity of computation, while the complexity of computation is $\Omega(n^{2.81})$ in the worst case. Besides, the proposed method can manage the priority of jobs for assigning the resources. As future work, we have a plan to use fuzzy-AHP in order to enhance the accuracy of the proposed method.

References

[1] Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008). Cloud computing and grid computing 360-degree compared. *arXiv preprint arXiv:0901.0131.*

[2] Cobham, A. (1954). Priority assignment in waiting line problems. *Journal of the Operations Research Society of America, 2*(1), 70-76.

[3] Phipps Jr, T. E. (1956). Machine repair as a priority waiting-line problem. *Operations Research, 4*(1), 76-85.

[4] Kleinrock, L. (1964). Analysis of A time-shared processor. *Naval research logistics quarterly, 11*(1), 59-73.

[5] Coffman Jr, E. G., & Kleinrock, L. (1968, April). Computer scheduling methods and their countermeasures. In *Proceedings of the April 30--May 2, 1968, spring joint computer conference* (pp. 11-21). ACM.

[6] Lee, Y. H., Leu, S., & Chang, R. S. (2011). Improving job scheduling algorithms in a grid environment. *Future generation computer systems, 27*(8), 991-998.

[7] Lee, M. C., Lin, J. C., & Yahyapour, R. (2015). Hybrid job-driven scheduling for virtual mapreduce clusters. *IEEE Transactions on Parallel and Distributed Systems, 27*(6), 1687-1699.

[8] Hwang, K., Dongarra, J., & Fox, G. C. (2018). *Cloud Computing and Distributed Systems: From Parallel Processing to the Internet of Things.* Morgan Kaufmann.

[9] Marozzo, F., Carretero Pérez, J., Duro, R., García Blas, J., Talia, D., & Trunfio, P. (2016). A data-aware scheduling strategy for dmcf workflows over hercules.

[10] Baranowski, M., Bubak, M., & Belloum, A. (2015, July). Data and process abstractions for cloud computing. In *2015 International Conference on High Performance Computing & Simulation (HPCS).* (pp. 646-649). IEEE.

[11] Aghababaeipour, A., & Ghanbari, S. (2018, February). A New Adaptive Energy-Aware Job Scheduling in Cloud Computing. In *International Conference on*

Soft Computing and Data Mining (pp. 308-317). Springer, Cham.

[12] Ghanbari, S., & Othman, M. (2018). Time Cheating in Divisible Load Scheduling: Sensitivity Analysis, Results and Open Problems. *Procedia Computer Science, 125,* 935-943.

[13] Singh, S., & Chana, I. (2015). QRSF: QoS-aware resource scheduling framework in cloud computing. *The Journal of Supercomputing, 71*(1), 241-292.

[14] Suresh, S., Huang, H., & Kim, H. J. (2015). Scheduling in compute cloud with multiple data banks using divisible load paradigm. *IEEE Transactions on Aerospace and Electronic Systems, 51*(2), 1288-1297.

[15] Ghanbari, S., & Othman, M. (2012). A priority based job scheduling algorithm in cloud computing. *Procedia Engineering, 50*(0), 778-785.

[16] Kong, X., Lin, C., Jiang, Y., Yan, W., & Chu, X. (2011). Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction. *Journal of network and Computer Applications, 34*(4), 1068-1077.

[17] Ficco, M., Di Martino, B., Pietrantuono, R., & Russo, S. (2017). Optimized task allocation on private cloud for hybrid simulation of large-scale critical systems. *Future Generation Computer Systems, 74,* 104-118.

[18] Narman, H. S., Hossain, M. S., Atiquzzaman, M., & Shen, H. (2017). Scheduling internet of things applications in cloud computing. *Annals of Telecommunications, 72*(1-2), 79-93.

[19] Wang, W., Zeng, G., Tang, D., & Yao, J. (2012). Cloud-DLS: Dynamic trusted scheduling for Cloud computing. *Expert Systems with Applications, 39*(3), 2321-2329.

[20] Xu, B., Zhao, C., Hu, E., & Hu, B. (2011). Job scheduling algorithm based on Berger model in cloud environment. Advances *in Engineering Software, 42*(7), 419-425.

[21] Juarez, F., Ejarque, J., & Badia, R. M. (2018). Dynamic energy-aware scheduling for parallel task-based application in cloud computing. *Future Generation Computer Systems, 78,* 257-271.

[22] Ghanbari, S., Othman, M., Bakar, M. R. A., & Leong, W. J. (2016). Multi-objective method for divisible load scheduling in multi-level tree network. *Future Generation Computer Systems, 54,* 132-143.

[23] Saaty, T. L. (1990). How to make a decision: the analytic hierarchy process. *European journal of operational research, 48*(1), 9-26.

[24] Saaty, T. L. (2013). The modern science of multicriteria decision making and its practical applications: The AHP/ANP approach. *Operations Research, 61*(5), 1101-1118.

[25] Ghanbari, S., Othman, M., Bakar, M. R. A., & Leong, W. J. (2015). Priority-based divisible load scheduling using analytical hierarchy process. *Applied Mathematics & Information Sciences, 9*(5), 2541.

[26] Strassen, V. (1969). Gaussian elimination is not optimal. *Numerische mathematik, 13*(4), 354-356.

Submitted 02.04.2019
Accepted 30.05.2019