JACET Journal of Advances in Computer Engineering and Technology, 3(3) 2017

Task Scheduling Algorithm using Covariance Matrix Adaptation Evolution Strategy (CMA-ES) in Cloud Computing

Ghazaleh Emadi[1], Amir Masoud Rahmani[2], Hamed Shah Hosseini[3]

Received (2017-03-03)
Accepted (2017-07-12)

***Abstract* - The need for planning the scheduling of the user's jobs has emerged as an important challenge in the field of cloud computing. It is mainly due to several reasons, including ever-increasing advancements of information technology and an increase of applications and user needs for these applications with high quality, as well as, the popularity of cloud computing among user and rapidly growth of them during recent years. This research presents the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), an evolutionary algorithm in the field of optimization for tasks scheduling in the cloud computing environment. The findings indicate that presented algorithm, led to a reduction in execution time of all tasks, compared to SPT, LPT, RLPT, GA and PSO algorithms.**

Keywords: **Cloud Computing, Task Scheduling, Virtual Machines (VMs), Covariance Matrix Adaptation Evolution Strategy (CMA-ES)**

1- Science and Research Branch, Islamic Azad University, Tehran, Iran.
2- Department of Computer Engineering Science and Research Branch, Islamic Azad University, Tehran, Iran. (rahmani74@yahoo.com)
3- Science and Research Branch, Islamic Azad University, Tehran, Iran.

1- Introduction

Nowadays, the internet and other associated technologies have turned out to be an integral part of human life, so that the need for information security, accessibility, easy information transportation, rapid processing of information, thwarting time-wasting and cost saving measures are so important and essential. Accordingly, a novel technology was devised, called cloud computing. The cloud computing is considered as a computational model which provides the clients with software, information, and resources upon any demand and need. The cloud computing can be defined as a utilization of computerized technologies and information via the internet. Indeed, the users can pay the charge and have access to the required information and resources (pay-as-per-use) offered by cloud computing using the internet or the network. The scheduling problem must be regarded as an important problem in the scope of cloud computing. So that, the cloud service providers can benefit the resources and clients should be able to access the needed application by paying a low fee [1]. Another issue which deserves attention is related to the execution mode of tasks, as the execution sequence of some tasks is important and in some cases, a single resource is capable of executing a specific task.

In fact, scheduling in cloud computing can be explained as follows: First, a client submits a request/job to get a service, and upon the arrival of the client's request, the service/resource provider triggers the scheduling of that request. That is, the resources are allocated to the jobs in

a manner that not only the efficient utilization of the computational resources is realized, but also the equality is established between the execution of applications. As a job consists of multiple tasks, the task scheduling notion in the context of cloud computing is highly sophisticated and complex [1]. Therefore, scholars have conducted numerous researches with the main focus on cloud computing and its task scheduling aspect. Our main objective of accomplishing this research is to present a method for reducing the response time of the task and increasing the processing power of cloud computing using Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [1]. As we reviewed the available literature on task scheduling algorithms in the cloud computing, we noticed that numerous algorithms have been presented by methods, such as particle swarm optimization (PSO) [15], Genetic algorithm [12-13], Fuzzy [14] and Simulated Annealing [16] and other heuristic and hybrid algorithms. However, we should point out that to date no study has been carried out in the field of using CMA-ES for task scheduling in cloud computing environment.

CMA-ES is an optimized evolutionary algorithm which samples a normal distribution and generates a new population. This evolutionary algorithm estimates a covariance matrix and means vector regarding the population. The various updating rules generate an adapted covariance matrix in each generation which plays an important role in the quality of the new population and result of the evolution direction. Hence, it is essential to study how the matrix being updated. On the other hand, nonparametric inter-class scatter matrix resulting from Fisher's linear discriminant (Fisher's classification method) [2] generates a pattern consisted of individuals with a high and low performance in the population. Then, it draws a vector from each group to the nearest neighbor k. This kind of vector points to a direction from which the individuals with low performance are led toward the proper performance. Consequently, this direction can be useful for better propelling of the population toward the optimal target [2].

The rest of this paper is organized as follows: Section 2 gives a brief outline of Literature Review on task scheduling problem. Section 3 introduces our algorithm for task scheduling in a cloud environment. Section 4 describes the simulation results. Finally, we conclude the paper in section 5.

2- Literature Review

In [1] an improved cost-based algorithm was presented for task scheduling process. This algorithm was suggested for efficient mapping of the tasks to the available resources in a cloud environment. Two main phases of the proposed algorithm involve using improved Bee Nest Algorithm for giving priority to the tasks and utilization of an algorithm for tasks grouping by their given priority. The scheduling mentioned above algorithm computes the fee paid for taking the resources and the efficiency of the calculations accomplished for completion of workflow tasks. A significant enhancement was observed during the investigation of the ratio of the paid fee for allocation of the resources to the efficient communication fee for the accomplishment of the workflow task.

Loshchilov et. Al. [2] presented time-cost balancing algorithm whose performance was studied by considering the cloud computing properties. For instance, the compression of workflow is carried out for reducing the execution time and cost by the input information provided by the user for the system during each login session. Another cost improvement algorithm for scheduling the workflow in hybrid clouds was presented in [3] which possesses two main phases, including tasks selection and resource selection from the public cloud, and finally the development of hybrid cloud. While the scheduler decides which task leads to the reduction in execution time using resources of public cloud, it should be noted that efficiency determination and execution costs play an important role during a new scheduling.

Paper [4] presented a genetic algorithm-based dynamic load balancing strategy. The speed of scheduling increases and switching between processors decreases, the tasks for execution are chosen, and in an advanced mode, the tasks' attributes are identified. The compatibility threshold contributes to the balancing of dynamic load of the processors. In [5], a fuzzy genetic optimization algorithm was recommended for scheduling the job to improve the resources in Hadoop framework (Apache Hadoop). A series of revisions was applied to the scheduling algorithm, and it estimates the execution time of the tasks for establishing a better load balance among the nodes of the cloud. However, selection of task

vector has a great impact on the prognosticated effectiveness.

Paper [6] suggested a particle swarm optimization algorithm (PSO) for task scheduling in cloud environment which minimizes the processing time. This algorithm has a more rapid convergence with a greater set of tasks. The lack of PSO variety leads to the false convergence. Therefore, the hybrid PSO along with various options are adopted for preventing from any false convergence during tasks scheduling. The present paper is focused on minimizing the finish time of jobs and maximizing the resources exploitation.

The main objective of a study carried out by [7] was to find an economic and optimal solution to the load distribution problem and to come up with planning for a proper allocation of energy resources cycled between thermal and wind generators which are available for service demand. Since the random behavior of wind speed and wind energy can be expressed using Probability Density Function (PDF). As a result, the scholars intended to minimize the expenses of the energy offered by conventional thermal generators and wind generators via presenting a framework by CMA-ES. To demonstrate the effectiveness and feasibility of the presented framework, they conducted various case study researches for these two thermal and wind generator systems. It should be declared that the security is highly crucial and vital for the workflow of various applications with bulky data and their execution and distribution over the infrastructure requires a longer time.

In [8], a secure and cost aware scheduling algorithm was introduced for heterogeneous jobs of workflow in a cloud environment. The suggested algorithm utilizes PSD-based Meta-heuristic optimization method for minimizing the total cost of workflow execution, whereas it takes into account the timeout constraint and risk rate. Different experiments were conducted using real world two workflow applications in CloudSim Simulation Framework, and the obtained results proved the effectiveness and practicality of the algorithm. Nowadays, as the cloud computing is employed as the infrastructure for distribution and execution of sophisticated and more heterogeneous applications, the scientists encounter with various conflicting and competing goals, such as decreasing the jobs computation time and economic expenses associated with the cloud platform.

In [9], a well-known scheduling algorithm, called HEFT was presented for dealing with multiple and conflicting goals of planning and workflow scheduling. To assess the novel algorithm regarding performance and cost, it was used for scheduling of artificial intelligence software and in the real world it was used in computations related to the Federal Clouds Infrastructure.

In [17] they have introduced an Optimized Task Scheduling Algorithm which adapts the advantages of various other existing algorithms according to the situation while considering the distribution and scalability characteristics of cloud resources.

Rezaque et al. [18] presented an algorithm for effective job scheduling. This algorithm provides a schedule of dividable task considering the network bandwidth which allows the allocation of workflow on the basis of availability of the network bandwidth. The proposed task scheduling algorithm utilizes a non-linear programming model for scheduling the dividable task which allocated a proper number of tasks to each virtual machine. Finally with regard to the afore-mentioned allocation mode, they devised an algorithm of scheduling for the dividable load considering the network bandwidth.

3- Proposed Algorithm

We assume that there exist i tasks and j virtual machines in the suggested system. Pij denotes the processing time of i-th task on the j-th Virtual Machine (VM) and Si1i2j represents the preparation time (communication cost) between i1 and i2 tasks on j-th virtual machine. You did not use Si1i2j!!!

Moreover, the main assumptions of the problem are as follows:

1- All tasks must be completed.

2- Each task can be executed and finished merely by one VM.

3- The start time of each task is equal to the sum of finish time (completion time) of the previous task and preparation time of the virtual machine.

4- The finish time of each task is equal to the sum of the start time of the task and its processing time.

5- The processing time of each task and

preparation time are randomly determined.

6- The allocation of virtual machines to the tasks is realized in a random manner, so there is a probability of idle state for some VMs during the distribution of tasks.

If it is supposed that the number of tasks is equal to 100 (I=100) and the number of VMs is equal to 20 (J=20). First, a 100×20 matrix of random integers was generated. The matrix represents the processing time (Pij) of i-th task on the VMs (j=1,2,…,20). Then, 100×100×20 matrix of random integers should be created on the various VMs for preparation time between various tasks. By using a random permutation of (I+J-1) length, the tasks will be distributed among the VMs. After that, the index of tasks related to each VM should be determined.

Thus, we will do as below:

1- Creation of matrix with such content: the first component of the matrix will be initialized by zero, and other components will be equal to separators index. Finally, one will be added to the matrix, that is:

$$\text{From} = [0\ 4\ 7\ 11] + 1 = [1\ 5\ 8\ 12] \quad (1)$$

2- Creation of matrix with such content: the initial components include separators index, and the end component is equal to I+J. Finally, one unit subtracted from matrix content, that is:

$$\text{To} = [4\ 7\ 11\ 14] - 1 = [3\ 6\ 10\ 13] \quad (2)$$

So, 'From' matrix is related to the first index of each VM and 'To' matrix is related to the last index of each VM.

So far, we applied the first and second assumptions. For the third and fourth assumptions following steps will be performed:

1- the start time of tasks

If it is the first task of a VM, its start time will be set to 0.

$$Start\_Time_i = 0 \quad (3)$$

Else, its start time will be equal to the finish time (finish time) of the previous task plus preparation time between (i-1)-th and i-th tasks on the same VM:

$$Start\_Time_i = Finish\_Time_{i-1} + Preparation\_Time_{i-1} \quad (4)$$

2- the finish time of each task is equal to the sum of the start time of that task and its processing time:

$$Finish\_Time_i = Start\_Time_i + Processing\_Time_i \quad (5)$$

3- VM finish time is equal to the finish time of the final task assigned to that VM:

$$VM\_Finish\_Time = (Finish\ time)_{final\_task} \quad (6)$$

4- Finish time of all tasks submitted to a cloud system is equal to the maximum of VM finish time:

$$System\_Finish\_Time = \max (VM\_Finish\_Time) \quad (7)$$

Regarding the fact that, the main objective of the present research is the realization of reduction in System_Finish_Time using CMA-ES, so suggested model of the allocation mode of tasks to the VMs and system time calculations will be applied as an input to the algorithm. To utilize the CMA-ES, a series of initial configurations for the parameters is required which will be discussed in what follows. For performing the simulation of the CMA-ES, we used the Pseudocode presented in [10].

As we assumed that the number of tasks is equal to 100 and number of VMs is equal to 20, thereby the length of random permutation is equal to 119.

1- We assumed that the maximum iteration number is equal to 250, the high and low limits of the values are equal to 30 and 2, respectively. Also, the values of Pc and Pσ , respectively, for updating of the anisotropic evolution path and isotropic evolution path are initialized by zero. The covariance matrix was assumed to be unique. Sigma is equal to multiply a random number and difference of low and high limits, which is used for updating the step size.

$$\begin{aligned} &MaxIteration=250 \\ &Min=2 \\ &Max=30 \\ &P\sigma=0 \\ &Pc=0 \\ &C=I \\ &Sigma=rand(1)\times(Max-Min) \end{aligned} \quad (8)$$

2- The population size (number of children) is calculated by the formula (9) [10]:

$$\lambda = 4 + \lfloor 3\log n \rfloor \times 10 \quad (9)$$

Where n is equal to the length of random permutation (n=119) and λ is the number of the iteration.

3- The parent number (or scores for recombination) which is equal to the half of the population size is calculated by formula 10 (i.e. 72 parents).

$$\mu' = \frac{\lambda}{2} \quad (10)$$

4- The parents' weights are calculated using formula (11), then the result will be normalized using formula (13).

$$\omega_i' = \log(\mu' + 0.5) - \log i \quad \text{for } i=1,\ldots,m \quad (11)$$

$$\mu = \lfloor \mu' \rfloor \quad (12)$$

$$\omega_i = \frac{\omega_i'}{\sum_{j=1}^{\mu} \omega_j'} \quad (13)$$

5- The variance-effectiveness (number of useful solutions) calculated by formula (14), is equal to the square of the total weights plus the total squares of the weights.

$$\mu_{eff} = (\sum_{i=1}^{\mu} \omega_i)^2 + \sum_{i=1}^{\mu} \omega_i^2 \quad (14)$$

6- Setting the control parameters for step size by formulas (15) and (16)

$$c_\sigma = \frac{\mu_{eff} + 2}{n + \mu_{eff} + 5} \quad (15)$$

$$d_\sigma = 1 + 2\max(0, \sqrt{\frac{\mu_{eff} - 1}{n+1}} - 1) + c_\sigma \quad 16)$$

7- Configuring the strategy parameters (adaptation) in covariance matrix by formulas (17), (18) and

$$c_c = \frac{4 + \mu_{eff}/n}{n + 4 + 2\mu_{eff}/n} \quad (17)$$

$$c_1 = \frac{2}{(n+1.3)^2 + \mu_{eff}} \quad (18)$$

$$c_\mu = \min(1 - c_1, \alpha_\mu \frac{\mu_{eff} - 2 + 1/\mu_{eff}}{(n+2)^2 + \alpha_\mu \mu_{eff}/2}) \; with \; \alpha_\mu = 2 \quad (19)$$

After performing the above-mentioned phases, we applied the cost function and the proposed model as an input to CMA-ES. Then, by creating a loop starting from one to the maximum number of the iteration, and the creation of a step size for children and multiplying it by sigma, and applying the product on the current parents, a new population will be generated. Now, the new population will be considered as a suggested permutation.

We have considered the task scheduling in cloud computing environment as a permutation. So, operators are as follows: 1) swap, 2) reversion and 3) insertion and these three operators were also applied to our algorithm. Now, we will apply the result of these operators to the proposed model to estimate the cost. So far, a phase of the step size was traversed. Hence, the above-mentioned phases will be iterated λ times.

After that, two formulas (20) and (21) will be used for selection and recombination of the individuals for generation of a new population, and then we perform the simulation.

$$\langle y \rangle_w = \sum_{i=1}^{\mu} \omega_i y_{i:\lambda} \; where \; \sum_{i=1}^{\mu} \omega_i = 1, \omega_i > 0 \quad (20)$$

$$m \leftarrow m + \sigma \langle y \rangle_w = \sum_{i=1}^{\mu} \omega_i x_{i:\lambda} \quad (21)$$

For updating the step size, two formulas (22) and (23) are used. refers to the updating of isotropic evolution path and denotes the updating of Sigma. The action of updating of the covariance matrix is carried out by two formulas (24) and (25). is related to the updating of the anisotropic evolution path, and the updating of covariance matrix will be accomplished according to the formula (25).

$$p_\sigma \leftarrow (1-c_\sigma)p_\sigma + \sqrt{c_\sigma(2-c_\sigma)\mu_{eff}}\, C^{-\frac{1}{2}} \langle y \rangle_w \tag{22}$$

$$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\| p_\sigma \|}{E\,\| N(0,I) \|} - 1\right)\right) \tag{23}$$

$$p_c \leftarrow (1-c_c)p_c + h_\sigma \sqrt{c_c(2-c_c)\mu_{eff}} \langle y \rangle_w \tag{24}$$

$$C \leftarrow (1-c_1-c_\mu)C + c_1\left(p_c p_c^T + \delta(h_\sigma)C\right) + c_\mu \sum_{i=1}^{\mu} \omega_i y_{i:\lambda} y_{i:\lambda}^T \tag{25}$$

The above actions will be executed by a specified maximum iteration number and finally, the best method for allocating the tasks to the VMs to with the minimum cost will be obtained.

4- Simulation and Results

Please 1) define your Simulation tool and 2) how to create conditions of DAG?

Figure 1 illustrates the start, processing and the finish time of all tasks in the assigned VMs. Furthermore, the total execution time of the submitted tasks to the VMs and total execution time of the tasks in a cloud computing system are also presented.

As shown in figure 1, the finish time of some VMs is so longer, and another VMs are in idle state, and the remaining VMs have a shorter finish time. For example, during the whole execution time of the tasks in a cloud computing system, only one task was assigned to 8-th and 13-th VMs. While, in the same system, the 14-th VM was always busy during the task execution process. For overcoming this problem, CMA-ES is used. Considering figure 2, we were able to overcome the idle state problem of some VMs and prevent the occurrence of long busy time for others via balancing the load among various VMs. Likewise, the tasks execution time of the whole system was reduced from 297 to 95.

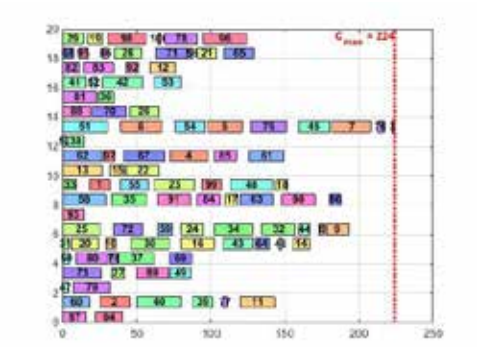

Figure 1: How tasks executed in the allocated machines along with execution time in the initial population

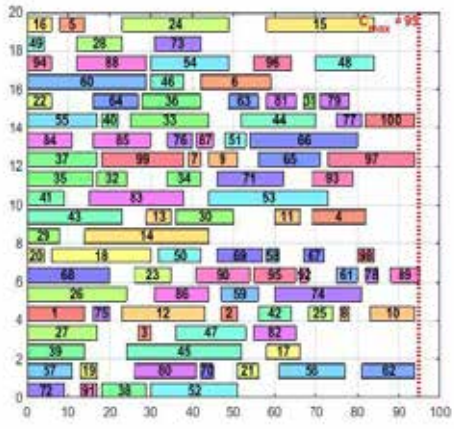

Figure 2: How tasks executed in the allocated machines along with execution time in the final population

Figure 3 depicts the cost reduction (the total System_Finish_Time) during consecutive iterations, and such reduction is more noticeable after 27-th iteration.

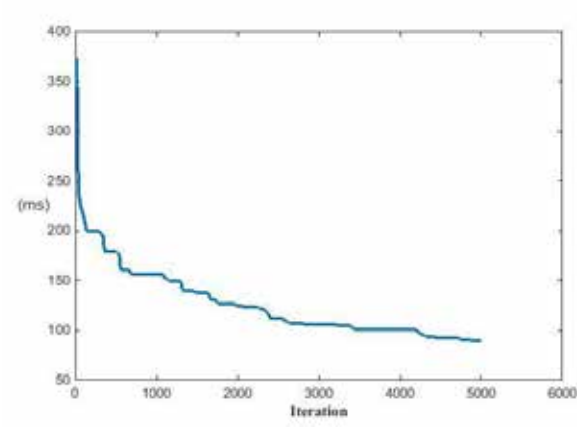

Figure (3): The best expenditure (the lowest time) during the iteration of simulation

For investing the performance of the proposed algorithm, we focused on the obtained results by modifying and manipulating the number of tasks and virtual machines. The number of tasks was varying between 60 to 180 and the number of virtual machines was set to 10, 20 and 30. With regard to figure (4), the tasks will be distributed among more machines, with increase in the number of virtual machines, and to the same ratio the time of task execution will decrease in the entire system.

For scrutinizing the performance of the suggested algorithm, the corresponding results will be compared to the non-preemptive Shortest Processing Time (SPT), Longest Processing Time (LPT) algorithms.

For assessing the performance of proposed algorithm, standards and metrics like parallelizing rate, exploitation of VMs and throughput are used by formulas (26) and (27):

$$\text{Parallelizing Rate} = \frac{\text{The total execution times}}{\text{The longest finish time of tasks}} \quad (26)$$

$$\text{Throughput} = \frac{\text{The total number of completed tasks per time unit}}{\text{Total time}} \quad (27)$$

The rate of parallelizing and operational power for the number of tasks and various virtual machines are presented in table (1).

For comparing the performance of the suggested algorithm with two non-preemptive algorithms, the same conditions were assumed in such a way that 100 tasks with a constant processing time along with 20 VMs were considered. Figures (5) and (7) illustrate that how tasks executed in allocated VMs along with their execution time for non-preemptive Shortest Processing Time (SPT) and Longest Processing Time (LPT) algorithms, respectively.

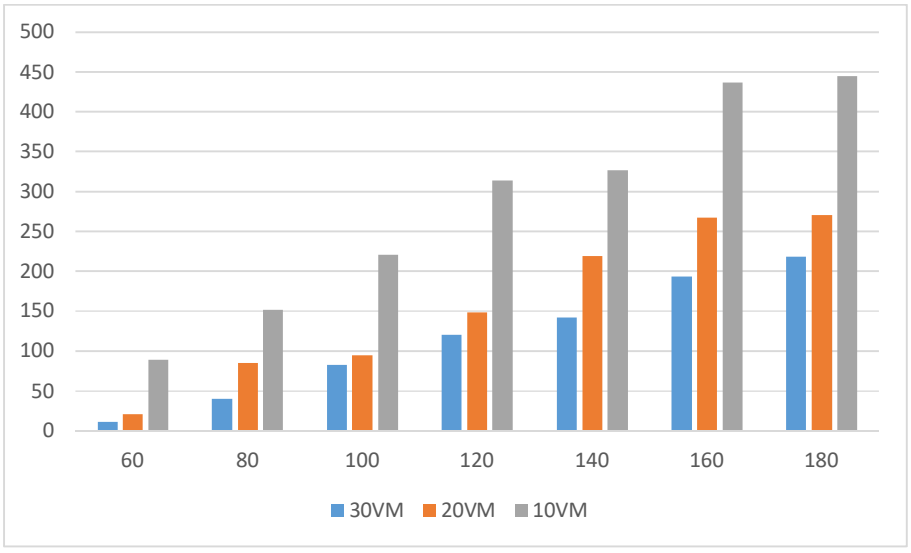

Figure (4): tasks' execution time of entire system for the number of tasks and various virtual machines

Table (1): Rate of parallelizing and operational power for the number of tasks and various virtual machines

		10	20	30
	60	18.2987	19.2834	24.1418
	80	18.1459	18.2741	24.0191
	100	18.0456	18.2315	23.9736
	120	17.9866	18.1912	23.9281
	140	17.9537	18.1432	23.8618
	160	17.7028	17.7321	23.7825
	180	17.5328	17.6891	23.7192
	200	17.4857	17.6511	23.6877
	60	0.0501	0.0526	0.0648
	80	0.0493	0.0512	0.0644
	100	0.0487	0.0508	0.0637
	120	0.0485	0.0504	0.0632
	140	0.0482	0.0497	0.0625
	160	0.0476	0.0492	0.0618
	180	0.0468	0.0486	0.0607
	200	0.0457	0.0479	0.0602

Table 2:Comparison of parallelizing rate, throughput of proposed algorithm with that of SPT and LPT algorithms

	SPT	LPT	Proposed
Parallelizing Rate	17.3548	18.1739	18.2315
Throughput	0.0465	0.0478	0.0508
Execution time of all tasks	124	115	95

Table3:Comparison of parallelizing rate, throughput of proposed algorithm with that of GA and PSO algorithms

	GA	PSO	Proposed
Parallelizing Rate	17.1867	18.2157	18.2315
Throughput	0.0497	0.0501	0.0508
Execution time of all tasks	109	103	95

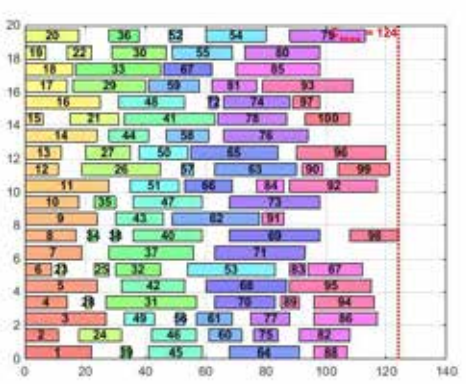

Figure 5: How tasks executed in the allocated machines along with execution time by SPT algorithm

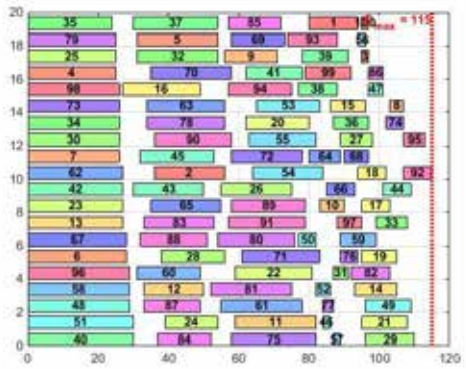

Figure 6: How tasks executed in the allocated machines along with execution time by LPT algorithm

Figure 7 shows that how tasks executed in the allocated VMs along with execution time for the proposed algorithm. It shows that the execution time of all tasks in suggested algorithm is less than that of non-preemptive SPT and LPT algorithms.

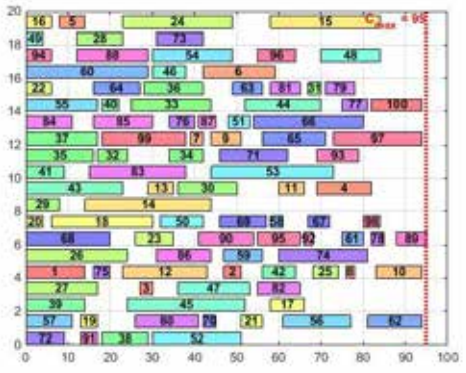

Figure 7: How tasks executed in the allocated machines along with execution time by proposed algorithm

Tables 2 presents the results of comparing the parallelizing rate and throughput of the proposed algorithm with the non-preemptive SPT and LPT algorithms. Table 2, shows that the parallelizing rate and throughput of the suggested algorithm are greater than two algorithms, SPT and LPT. Also, the total execution time of all tasks of the proposed algorithm is less than that of other algorithms. The parallelizing rate, throughput and execution time of all tasks associated with the proposed algorithm is equal to 18.2315, 0.0508 and 95, respectively.

In what follows, for further investigations of the proposed algorithm a comparison was made between GA and PSO in terms of performance and the obtained results of this comparison are presented in table (3). It can be noted that, the parallelizing and operational power rate of the proposed algorithm is greater than that of PSO and GA algorithms.

5- Conclusion

In this research, CMA-ES was presented as an algorithm in the field of optimizing the response time of the response to the user's request in a cloud computing environment and for simulating the preparation time, the execution time of the tasks were obtained randomly, then the VMs were allocated to the tasks on the basis of each task's priority using the proposed model. It was assumed that the model specifies the order of tasks execution in each machine and it stores the order in an array, we separated the tasks of various machines using a separator. Then, the array was taken into account as an initial population and applied as an input to the CMA-ES. This strategy estimates a covariance matrix and means vector from the population. The various updating rules fabricate an adapted covariance matrix in each generation, which plays a vital role in the quality of the new population and directing the evolution. we commenced simulation, and we were able to reduce the execution time of all tasks of the system significantly. For appraising the performance of the proposed algorithm, we compared it with two non-preemptive algorithms, including SPT and LPT algorithms.. While the finish time of the proposed algorithm is equal to 95. Generally speaking, the performance of the proposed algorithm is better than of non-

preemptive algorithms.

On the other hand, our further comparison results indicated that the parallelizing and operational power rate associated to the proposed algorithm is greater than that of PSO and GA algorithms.

REFERENCES

[1] Choudhary M., Peddoju S.K.,(2012). A Dynamic Optimization Algorithm for Task scheduling in Cloud Environment, International Journal of Engineering Research and Applications (IJERA), Vol. 2, Issue 3, pp. 2564–2568.

[2] Loshchilov, Ilya, et al. "Maximum likelihood-based online adaptation of hyper-parameters in CMA-ES." International Conference on Parallel Problem Solving from Nature. Springer International Publishing, 2014.

[3] Mell P., Grance T., (2013). "The NIST Definition of Cloud Computing":http://productionscale.com/blog/2011/8/7/the-nist-definition-of-cloud-computingdrft.html, [Accessed: 20-Dec-2013].

[4] Jorge Peñarrubia1 , Facundo A. Gómez , Gurtina Besla , Denis Erkal4 & Yin-Zhe Ma., (2015)., "A timing constraint on the (total) mass of the Large Magellanic Cloud"., Astrophysics of Galaxies.

[5] Agnetis.A, Billaut.Ch.J, Gawiejnowicz.S, Pacciarelli.D, Soukhal.A, (2014). Multiagent Scheduling, Models and Algorithms, Springer US .

[6] Liu.J, (2013). Job Scheduling Model for Cloud Computing Based on Multi-Objective Genetic Algorithm, International Journal of Computer Science Issues ISSN : 1694-0814 .

[7] Ghorbannia Delavar, A., Javanmard , M., Barzegar Shabestari and Marjan Khosravi Talebi ., (2012). "RSDC (RELIABLE SCHEDULING DISTRIBUTED IN CLOUD COMPUTING)" in International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.3, June 2012.

[8] M. Dakshayini, Dr. H. S. Gurupras.ad. (2011). "An Optimal Model for Priority based Service Scheduling Policy for Cloud Computing Environment" International Journal of Computer Applications (0975 – 8887) Volume 32– No.9.

[9] Shamsollah Ghanbari, Mohamed Othman. (2012). "A Priority based Job Scheduling Algorithm in Cloud Computing" International Conference on Advances Science and Contemporary Engineering.

[10] Hansen, N. (2011). The CMA Evolution Strategy: A Tutorial. June 28.

[11] Nagadevi.S, Satyapriya.K, Malathy.D. (2013). A Survey on Economic cloud schedulers for optimized task scheduling, International Journal of Advanced Engineering Technology, Vol 5, pp: 58-62.

[12] Jafari Navimipour, Nima, Amir Masoud Rahmani, Ahmad Habibizad Navin, and Mehdi Hosseinzadeh. "Job scheduling in the Expert Cloud based on genetic algorithms." Kybernetes 43, no. 8 (2014): 1262-1275.

[13] Rahmani, Amir Masoud, and Mojtaba Rezvani. "A novel genetic algorithm for static task scheduling in distributed systems." International Journal of Computer Theory and Engineering 1.1 (2009): 1.

[14] Adabi, Sahar, Ali Movaghar, and Amir Masoud Rahmani. "Bi-level fuzzy based advanced reservation of Cloud workflow applications on distributed Grid resources." The Journal of Supercomputing 67.1 (2014): 175-218.

[15] Dashti, Seyed Ebrahim, and Amir Masoud Rahmani. "Dynamic VMs placement for energy efficiency by PSO in cloud computing." Journal of Experimental & Theoretical Artificial Intelligence 28.1-2 (2016): 97-112.

[16] Kazem, Ali Asghar Pourhaji, Amir Masoud Rahmani, and Hamed Habibi Aghdam. "A modified simulated annealing algorithm for static task scheduling in grid computing." Computer Science and Information Technology, 2008. ICCSIT'08. International Conference on. IEEE, 2008.