Technical Journal, University of Engineering and Technology (UET) Taxila, Pakistan Vol. 22 No. 4-2017
ISSN:1813-1786 (Print) 2313-7770 (Online)

Scheduling Hard Real-Time Tasks in Cloud Computing Using Differential Evolution Algorithm

A. Mahmood[1], M. U. Ahmed[2]

[1]*Executive Director, Abasyn Universty (Islamabad Campus)*
[2]*Computer Science Department, AIOU, Islamabad*
[2]moiz.ahmed@aiou.edu.pk

Abstract-The cloud computing environment, in which virtual resources/machines are made available over the Internet, is a smart option for many real-time applications. However, there are a number of critical problems that need to be addressed to successfully use cloud computing for real-time applications. One of the challenges is the allocation and scheduling of real time tasks on the virtual machines. In this paper, we modeled the task allocation and scheduling problem as a binary optimization problem. The differential evolution (DE) algorithm has been customized to solve the problem. A detailed experimental study has been conducted to investigate the solution quality of DE algorithm. The results show that DE performs better than the greedy and the genetic algorithm (GA).

Keywords-Cloud Computing, Real-time Systems, Task Scheduling, Genetic Algorithm, Differential Evolution

I. INTRODUCTION

Cloud computing enables virtual access to shared computing over the Internet on the basis of usage cost [i] making it an appealing option for many organizations [ii - iii]. Cloud-computing services are offered in three standard models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [i]. In IaaS model, a service provider provides automated and scalable computing infrastructure by hosting applications and making them available to customers on the Internet.

A PaaS model offers hardware resources such as server & networking infrastructure and software resources such as operating system & application programs for the application development. The SaaS deals with the management and deployment of software through a virtual environment that can be rented by the users as per demands. This study focuses on IaaS model in which cloud resources can be subscribed in the form of Virtual Machines (VMs) to execute a number of real-time tasks. A VM provides an application environment comprising of a certain computational power, storage, and communication resources. The VMs may be heterogeneous, that is, they may have different computational capabilities. The cloud computing is a smart option for many real-time applications (e.g., object detection, navigation control, complex applications, monetary systems, etc.) [iv]. A real-time application may comprise of a number of tasks precedence constraint and deadlines, that is, a task can execute only after the completion a number of other tasks. A careful scheduling of tasks on VMs is important to utilize computing resources effectively [v]. This gives rise to a well-known NP-complete problem referred as task-allocation problem [vi].

This paper models the task allocation and scheduling as an optimization problem. The model captures both the processing and data transmission costs. The task precedence relationships and deadlines are considered as constraints. This paper adapts differential evolution (DE) algorithm for the problem. A two-dimensional topology preserving solution encoding scheme is used. The DE algorithm has been empirically studied and compared with a number of other algorithms for their relative solution quality.

The remaining paper is organized as follows. Literature review is presented in Section 2. Section 3 presents our system model and problem. The proposed algorithm is presented in Section 4. The performance and comparison of results is given in Section 5. Finally, conclusion is given in Section 6.

II. LITERATURE REVIEW

Numerous research studies on task allocation and scheduling in cloud-computing environment have been reported in the literature. These studies are primarily focused on optimized use of energy/power, make span, economic cost, and achieving the required level of quality of service (QoS) [vii - x]. A scheduling algorithm improvising QoS metrics, such as load balancing, average latency, and make span, have been

proposed by Wu et al. [viii]. In their approach, tasks (processes) having higher priority are allocated first. A multi-objective QoS workflow scheduling algorithm is proposed in [xi]. Jang et al. proposed a genetic algorithm to deal with task scheduling problem and maximize overall QoS [xii]. The profit maximization was addressed by Lee et al. [ix] who worked on maintaining adequate level of QoS by using scheduling algorithms.

An efficient task-scheduling algorithm based on non-linear programming method is proposed by Razaque et al. [xiii].They used the availability of network bandwidth for task scheduling and allocation to VMs. In [xiv], Min-Max and Min-Min algorithms are proposed. The algorithms proposed in [xv] by Li. et al. dynamically adjust resource allocation at the run time. A differential evolution algorithm to reduce make span and overall cost is proposed by Tsai et al. [xvi]. Cuckoo Search Algorithm, Bat algorithm and many other meta heuristics algorithms have also been proposed for the task allocation problem [xvii - xx]. All the work mentioned before does not explicitly models allocation and scheduling of real-time tasks.

Liu et al. [xxi] proposed a procedure for dealing with real-time tasks. The objective is to maximize the total utility in a non-preemptive scheduling environment. Their proposed algorithm showed better performance as compared to traditional scheduling algorithms and Earliest Deadline First (EDF) strategy. The real-time task allocation problem was addressed by Kumar et al.[iv]. The constrained optimization modeling was based on execution speed and cost on different VMs. The authors then proposed a temporal overlap using a greedy approach. They, however, did not consider the communication cost in their model. Deniziak [xxii] proposed the scheduling scheme of soft and real-time applications. The objective is to reduce the total cost by using genetic algorithm.

Unlike the previous work, our work is focused on minimizing total communication and processing costs for real time tasks. The operators of differential evolution algorithm have been specialized for the problem and two dimensional solution encoding scheme.

III. System Model and Problem

Suppose there are finite number of real-time tasks, $T = \{t_1, t_2,...t_N\}$ to be assigned and scheduled on M virtual machines. The computation speed (measured in clock cycles/unit time) and cost of a virtual machine m_k are denoted by s_k and c_k respectively. Each task $t_i \in T$ is characterized a pair (w_i,d_i) where w_i is workload (total clock cycles required) and d_i is the deadline. Task in T may have precedence relationship. The quantity of data transferred from t_i to t_j is denoted by v_{ij}. We further define $Pre(t_i) = \{t_j | t_j \in T, e_{ji} \in E\}$ and $Succ(t_i) = \{t_j | t_j \in T, e_{ij} \in E\}$ as sets of tasks that are immediate predecessors and are immediate successors of t_i, respectively. Also, $aPre(t_i) = \{t_j, t_k, t_l..., t_p\}$ is the set of predecessors of task t_i, if $\{e_{jk}, e_{kl}...e_{(p-1)p}, e_{pi} \in E \text{ and } Pre(t_j) = \phi\}$.

Now, the time required to execute task $t_i \in T$ on a virtual machine m_k, denoted by ET_{ik}, is given by [xx- xxiv]

$$ET_{ik} = \frac{w_i}{s_k} \quad (1)$$

Where s_k denotes is the speed of the VM m_k. The *Latest Start Time* (LST_i) of task t_i and *Earliest Start Time of task* t_i (EST_i) can be given by Equation (2) and (3), respectively [xx- xxiv].

$$LST_i = d_i - ET_{ik} \quad (2)$$

$$EST_i = \begin{cases} \max\limits_{j \in Pre(t_i)} \{FT_j\} \\ 0 \quad \text{if } Pre(t_i) = \phi \end{cases} \quad (3)$$

If ST_{ik} and Et_{ik} denote the actualstart time and processing time of task t_i on the VMm_k, respectively, then the finish/completion time of task $t_i(FT_i)$ is can be calculated by the following equation [xx]:

$$FT_i = ST_i + ET_{ik} \quad (4)$$

Since a cloud service providers charge a cost of C_k for leasing a VMm_k for at least L units of time in minutes and hours etc. irrespective of the actual usage, the execution cost of task t_i on m_k, in case no other task is reserved on the same VM, is given by

$$EC_{ik} = \left\lceil \frac{ET_{ik}}{L} \right\rceil \times C_k \quad (5)$$

The total cost of executing all the tasks is given

$$CC(X) = \sum_{i=1}^{N} \sum_{j \in Pre(t_i)} \sum_{k=1}^{M} \sum_{l=1}^{M} x_{ik} \times x_{jl} \times v_{ij} \times b_{kl} \quad (6)$$

Where X represents a matrix of MXN order whose entry $x_{ik} = 1$ if task t_i is booked on VMm_k, and $x_{ik} = 0$ otherwise. The parameter b_{kl} denotes the data transmission cost/unit data between VMssm_k and m_l. If two VMs reside on the same server, then data transmission cost is taken as zero

The total execution cost for the task set T is given by:

$$EC(X) = \sum_{i=1}^{N} \sum_{k=1}^{M} x_{ik} \times EC_{ik} \quad (7)$$

Total cost, TC (X), can be calculated by adding

$CC(X)$ and $EC(X)$, that is

$$TC(X) = CC(X) + TC(X) \quad (8)$$

The objective is finding an X that minimizes $TC(X)$ [xx - xxiv].

That is,

Minimize $TC(X) = CC(X) + TC(X)$
Subject to:

$$\sum_{k=1}^{M} x_{ik} = 1 \text{ for each } i,\ 1 \le i \le N \quad (9)$$

$$FT_i \le d_i \text{ for each } i,\ 1 \le i \le N \quad (10)$$

$$ST_i \ge EST_i \text{ for each } i,\ 1 \le i \le N \quad (11)$$

The limitation of reserving and scheduling each task on exactly one VM is represented by equation (9). The constraint (10) specifies that each task must meet its deadline and the last constraint (11) ensures that a task can start only after predecessor tasks.

IV. Proposed Algorithm

Differential Evolution (DE) is a population-based algorithm adapted to solve our problem [xxii]. The algorithm itself consists of three main operators, namely the mutation, crossover and selection. The mutation and crossover operators are used to produce new vectors (solutions), and the selection operator determines whether a vector can move to the next iteration or not. Since it is a population-based algorithm, the initialization of solutions is required to start executing. Fig. 1 shows the adapted DE algorithm. The solution encoding, generation of initial solution, and DE specific operators are discussed in the following sub-sections.

Step 1 Generate initial population
Step 2 Evaluate the fitness of each vector
Step 4 while (not termination condition) {
Step 5for each parent x_i^k {
Select the target vector x_3^k randomly
Select two distinct vectors x_1^k, x_2^krandomly
for each element *j* produce the trial vector v_i^{k+1} using

$$v_{ij}^{k+1} = x_{3j}^k + F(x_{1j}^k - x_{2j}^k)$$

}
Step 6 for each element *j* produce the offspring u_{ij}^{k+1} using

$$u_{ij}^{k+1} = \begin{cases} v_{ij}^{k+1} & if\ rand < CR \\ x_{ij}^k & Otherwise \end{cases}$$

Step 7 Replace x_i^k with u_{ij}^{k+1} if u_{ij}^{k+1} is feasible and better

}

Fig. 1. The proposed DE algorithm

A. Solution representation and initial population generation

In the proposed DE algorithm, a solution is embodied by a two dimensional array. The first row of a chromosome represents tasks in their topological order (from left to right) and the second row represents the corresponding virtual machine number on which a task is to be executed, as shown in Fig. 2. The topological ordering of tasks ensures that a schedule conforms to the task precedence constraint. Initial population is generated (step 1 of the DE algorithm) using the algorithm given in [xxiii].

Task	T0	T1	T2	T3	T4
VM	1	5	0	2	5

Fig. 2. A representation of schedules in terms of a chromosome.

B. Mutation Operator

For each parent x_i^k in generation *k*, a target vector x_3^k is selected along with two other vectors x_1^k and x_2^k belonging to the same generation *k*, with $i_1 \neq i_2 \neq i_3$ (Step 5 of the DE algorithm). The mutant vector v_1^{k+1} is calculated at step 5 using the following formula:

$$v_i^{k+1} = x_3^k + F(x_1^k - x_2^k) \quad (13)$$

Where, F ($(F \in [0, 2])$ is the mutation factor used to control the amplification of the differential variation. Fig. 3 shows an example of producing a mutant vector. Since our problem is a discrete optimization problem, we take only the absolute integer part of each v_{ij}^{k+1} to denote the virtual machine number.

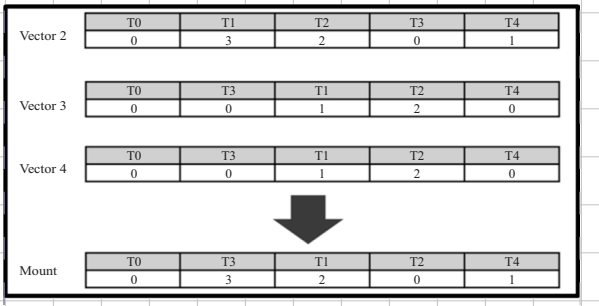

Fig. 3. DE mutation performed on three distinct vectors assuming F=0.5

C. Crossover operator

The crossover operator combines elements from parent x_i^k and the mutant vector v_i^k in order to create an offspring u_i^k (Step 6). This step is applied with a probability *CR* using the following formula:

$$u_{ij}^{k+1} = \begin{cases} v_{ij}^{k+1} & if\ rand < CR \\ x_{ij}^{k} & Otherwise \end{cases} \quad (14)$$

D. Selection Operator

If the offspring u_{ij}^{k+1} is feasible, its fitness is evaluated and compared with the fitness value of the parent vector x_i^k. If the fitness of offspring u_{ij}^{k+1} is higher than that of parent's fitness value, then offspring u_{ij}^{k+1} replaces the parent (Step 7). That is

$$x_i^{k+1} = \begin{cases} u_i^{k+1} & if\ fitness(u_i^{k+1}) < fitness(x_i^k) \\ x_i^k & Otherwise \end{cases} \quad (15)$$

V. Simulation Results and Discussions

The performance of the adapted DE algorithm was investigated and its performance was compared with standard genetic algorithm and greedy algorithm [xxiv]. All the algorithms were programmed using C++ and executed on computers with Intel *i*7 processor, 8 GB RAM and Microsoft Windows 10 platform.

As per other studies a standard approach of data generation was adopted. The set of tasks to be scheduled were generated as DAGs using the TGFF utility [xxiv]. The number of tasks in a DAG was selected randomly between 10 and 300. The workload were generated for the tasks within the interval of [10, 4500] as used in a previous study [iv]. The tasks were also assigned deadlines using the technique suggested in [xxv]. The volume of data exchanged between tasks was randomly generated and was in the range 50 - 1500. The total number of accessibleVM was 50. Different set of values were assigned randomly for cost and speed of the virtual machine as given in [iv, xvi, xxvi]. The transmission costs/unit data between the VM were also generated in the range 1 - 5.

The greedy algorithm for each input combination was executed once. However the genetic algorithm and the DE algorithms were run for 30 times for each input combination. This is a standard practice for analyzing the performance of iterative heuristics. The parameters used for GA and DE are given in Table I.These parameters were selected after performing parameter sensitivity analysis for both GA and DE for DAGs having 5, 10 and 15 tasks each.

TABLE I
Parameter Settings for GA and DE

Algorithm	Parameter Setting
Genetic Algorithm	Population size: 40 Parent selection: Roulette-wheel Crossover rate: 0.8 Mutation rate: 0.05
Differential Evolution	Population size: 30 Mutation factor: 0.5 Crossover rate: 0.8

Table II give value of the cost function obtained by each algorithm. For DE and GA, the average cost for 30 independent run is reported. The results show that performance of the greedy algorithm is the worst as compared to GA and DE for all the test cases. It can also be observed that DE outperforms both the greedy as well as GA algorithm in terms of quality of the solution. The Wilcox on matched pair test indicates that solution quality of DE algorithms is statistically significant. In addition, the results also showed that DE's performance is more stable among different independent runs as is evident from with a low standard deviation value as compared to that of GA. The percentage improvement of DE over greedy and GA is shown in Table III. The results indicate that the improvements achieved by DE over the greedy algorithm were extensively high 16.65% to 45.87%. The percentage improvements of DE over GA were in the range of over 5.17% to almost 15.99%. The results indicated that all improvements achieved by DE were statistically significant.

TABLE II.
Comparison of Costs of Greedy, GA, and DE Algorithms

No. of Tasks	Greedy	GA	DE
10	41.28	30.52	28.30
20	94.13	79.35	75.45
40	210.24	162.71	149.26
80	438.82	345.60	316.27
100	480.91	405.35	349.47
200	689.62	632.15	591.20
300	1015.36	892.72	834.75

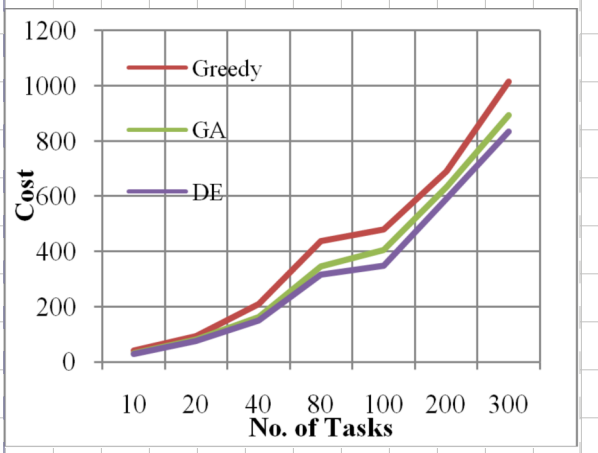

Fig. 4. Graph plotted using No. of Tasks and Cost of Algorithms

TABLE III
PERCENTAGE IMPROVEMENT OBTAINED BY DE OVER GREEDY, AND GA

Tasks	DE vs. Greedy	DE vs. GA
10	45.87	7.84
20	24.76	5.17
40	40.85	9.01
80	38.75	9.27
100	37.61	15.99
200	16.65	6.93
300	21.64	6.94

V. CONCLUSION AND FUTURE WORK

Task allocation and scheduling on virtual machines in a cloud computing environment is a well-known NP-hard problem. This paper presented the differential evolution algorithm customized for solving the task allocation and scheduling problem. The solution encoding scheme and various operators of DE are presented. The experimental results show the performance of DE is significantly better than the performances of the greedy as well as GA algorithms in terms of solution quality. As a future work, we intend to test the performance of the DE algorithm with other population based algorithms. Developing hyper heuristics can also be another future direction.

REFERENCES

[i] J. Z. Luo, J. H. Jin, A. B. Song, and F. Dong, "Cloud Computing: Architecture and Key Technologies", Journal of China Institute of Communications, Vol. 32, No. 7, pp.3-21, 2011.

[ii] M. Abdullahi, M. Ngadi, "Hybrid Symbiotic Organisms Search Optimization Algorithm for Scheduling of Tasks on Cloud Computing Environment", PLoSone, Vol. 11, pp. 6-26, 2016

[iii] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, M. Zaharia, "A view of cloud computing", Communications of the ACM, Vol. 53, No. 4,pp. 50-58, 2010.

[iv] K. Kumar, J. Feng, Y. Nimmagadda, Y. H. Lu, "Resource allocation for real-time tasks using cloud computing", Proceedings of IEEE 20th International Conference on Computer Communications and Networks (ICCCN), Maui, HI, USA, pp. 1–7, 2011.

[v] A. I. Awad, N. A. El-Hefnawy, H. M. Abdelkader, "Enhanced Particle Swarm Optimization for Task Scheduling in Cloud Computing Environments", Procedia Computer Science, Vol. 65, pp. 920-929, 2015.

[vi] N. J. Navimipour, L. M. Khanli, "The LGR method for task scheduling in computational grid", Proceedings of International Conference on Advanced Computer Theory and Engineering, Phuket, Thailand, pp. 20-22 pp. 1062–1066, 2008.

[vii] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, J. Wu, "Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment", Journal of Systems and Software, Vol. 99, 20–35, 2015.

[viii] X. Wu, M. Deng, R. Zhang, B. Zeng, S. Zhou, "A task scheduling algorithm based on QoS-driven in cloud computing", Procedia Comput. Sci. Vol. 17, pp. 1162–1169, 2013.

[ix] Y. C. Lee, C. Wang, A. Y. Zomaya, B. B. Zhou, "Profit-driven scheduling for cloud services with data access awareness", Journal of Parallel and Distributed Computing , Vol. 72, No. 4, pp. 591-602, 2012.

[x] S. K. Panda, I. Gupta, P. K. Jana, "Allocation-aware Task Scheduling for Heterogeneous Multi-cloud Systems", Procedia Computer Science, Vol. 50, pp. 176-184, 2015.

[xi] A. Sangwan, G. Kumar, S. Gupta, "To Convalesce Task Scheduling in a Decentralized Cloud Computing Environment, Review of Computer Engineering Research,Vol. 3, No. 1, pp. 25-34, 2016

[xii] S. Jang, T. Kim, J. Kim, J. Lee, "The study of genetic algorithm-based task scheduling for cloud computing", International Journal of Control and Automation, Vol. 5, No. 4, pp. 157-162, 2012.

[xiii] A. Razaque, N. Vennapusa, N. Soni, G. Janapati, "Task scheduling in Cloud computing", Proceedings of IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, USA, pp.1-5, 2016.

[xiv] S. Sindhu, "Task scheduling in cloud computing", International Journal of Advanced Research in Computer Engineering & Technology, 2015, Vol. 4, pp. 3019–3023, 2015.

[xv] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, Z. Gu, "Online optimization for scheduling preemptable tasks on IaaS cloud systems", Journal of Parallel and Distributed Computing, Vol. 72, No. 5, pp. 666-677, 2012.

[xvi] J. T. Tsai, J. C. Fang, J. H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm", Computers & Operations Research, Vol. 40, No. 12, pp. 3045-3055, 2013

[xvii] S. Pandey, L. Wu, S. Guru, R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments", Proceedings of 24th IEEE International Conference on Advanced

Information Networking and Applications (AINA), Perth, Australia, pp. 400–407, 2010.

[xviii] N. Navimipour, f. Milani, "Task scheduling in the cloud computing based on the cuckoo search algorithm", International Journal of Modeling and Optimization 5, Vol. 5, no. 1, pp. 44, 2015

[xix] S. Raghavan, P. Sarwesh, C. Marimuthu, K. Chandrasekaran, "Bat algorithm for scheduling workflow applications in cloud", Proceedings of International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV), Shillong, India, pp. 139-144, 2015.

[xx] R. A. Bahlool, "Optimized Real-time Tasks Allocation in Cloud Computing Environment", MSc Thesis, University of Bahrain, 2016

[xxi] S. Liu, g. Quan, s. Ren, "On-line scheduling of real-time services with profit and penalty" Proceedings of ACM Symposium on Applied Computing, Taichung, Taiwan, pp. 1476-1481, 2011.

[xxii] S. Deniziak, L. Ciopinski, G. Pawinski, K. Wieczorek, S. Bak, "Cost optimization of real-time cloud applications using developmental genetic programming", Proceedings of IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC), London, UK, pp.774-779, 2014.

[xxiii] K. V. Price, R. M. Storn, J. A. Lampinen,."The differential evolution algorithm", Differential evolution: a practical approach to global optimization, pp.37-134, 2005.

[xxiv] A. Mahmood, S. A. Khan, (2017),"Hard Real-Time Task Scheduling in Cloud Computing Using an Adaptive Genetic Algorithm", Computers, Vol. 6, No. 2, pp.15, 2017.

[xxv] R. P. Dick, D. L. Rhodes, W. Wolf, "TGFF: Task graphs for free" Proceedings of 6th International Workshop on Hardware/software Codesign, Seattle, WA, USA, pp.97-101, 1998.

[xxvi] C. W. Tsai, W. C. Huang, M. H. Chiang, M. C. Chiang, C. S. Yang, "A hyper-heuristic scheduling algorithm for cloud", IEEE Transactions on Cloud Comput. Vol. 2, pp. 236-250, 2014