A new SLA-aware Load Balancing Method in the Cloud using an Improved Parallel Task Scheduling Algorithm

Mehran Ashouraei
Young Researchers and Elite Club
Tabriz Branch, Islamic Azad University
Tabriz, Iran
m.ashouraei@gmail.com

Seyed Nima Khezr, Rachid Benlamri
Department of Software Engineering
Lakehead University
Thunder Bay, Canada
{skhezr, rbenlamr}@lakeheadu.ca

Nima Jafari Navimipour
Department of Computer Engineering
Tabriz Branch, Islamic Azad University
Tabriz, Iran
jafari@iaut.ac.ir

***Abstract*—Cloud computing as a novel and entirely internet-based computing platform is emerging and its tenacious challenges become more vivid. A parallel genetic algorithm-based method for scheduling tasks with priorities is provided in this paper. The goal is to efficiently utilize resources and reduce resource wastage in cloud environments. This is achieved by improving the load balancing rate while better resources are selected to fulfill arrival tasks in a shorter time with lower task failure rate. To evaluate the proposed method, it is simulated using Matlab and compared with two existing methods, a hybrid Ant colony-honey method and a Round-Robin (RR) based load balancing method. The results show that the proposed method has 9% - 31% lower energy usage, 14% - 37% lower migration rate and 13%- 17% better Service Level Agreement (SLA) in comparison with the Hybrid and RR method.**

Keywords—task scheduling; load balancing; parallel genetic algorithm; restriction table.

I. INTRODUCTION

Cloud computing as a novel and entirely internet-based approach provides a highly available, scalable, and flexible computing platform for a variety of applications and has brought about great benefits to both enterprises and individuals [1]. Computing is being changed to a service-based model whereby access to these services depend on users' requirements without regard to where the services are hosted or how they are delivered [2]. Such computing model offers many types of services, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). With the spread of cloud computing, cloud workflow systems are designed to facilitate the cloud infrastructure to support huge scale distributed collaborative e-business and e-science applications [3].

An ever-growing number of corporations, companies and individuals are running applications of all sorts in the cloud. Such a huge demand for services, at different scale, led to the development of sophisticated load balancing techniques [5] to efficiently utilize resources and reduce resource wastage in cloud environments [4]. One of the ways to balance the load is scheduling tasks in the cloud. Task scheduling is an important part of any distributed system like P2P networks, Grid and cloud [6]. It assigns tasks to suitable resources for execution. However, it is an NP-Complete to schedule tasks to obtain the least makespan [7]. The main goal of a task scheduling algorithm is scheduling all subtasks on a given number of accessible resources in order to minimize makespan without violating precedence constraints [8]. The most vital side of cloud computing environments is load balancing. Efficient load balancing scheme assures effective resource utilization by provisioning of resources to the cloud user's on-demand basis in pay-as-you-say-manner. Load balancing may even support prioritizing users by applying appropriate scheduling criteria [9]. As load balancing algorithms depend on current situation of system, it is considered a dynamic issue to balance its load.

Meta-heuristic approaches are inclusively applied to find optimal or near-optimal solutions for the scheduling problem. Owing to implicit parallelism and intelligence, genetic algorithms (GAs) have been applied to solve some large-scale, nonlinear resource scheduling [10] and good results have been obtained from their use in task scheduling. In this paper, we propose a genetic-based scheduling algorithm to assign resources to tasks. Our aim is to improve load balancing rate of GA while better resources are selected to fulfill arrival tasks in a shorter time with lower task failure rate. This is achieved by (i) improving the used genetic algorithm with a novel restriction table; (ii) considering SLA requirements of users; and (iii) reducing run-time by calculating fitness value of chromosomes in parallel.

The rest of this paper is organized as follows. Section 2 provides a review of related work. Section 3 deals with the detailed description of proposed method, while section 4 provides simulation results and a comparative study with similar methods. The last section concludes this paper and provides some future work.

II. RELATED WORK

The management and scheduling of resources in cloud environment is complex, and therefore demands sophisticated tools for analyzing scheduling algorithms before applying them to real systems [3]. There are many papers that address the problem of scheduling in distributed systems, like Grid and multiprocessor systems, whereas, there is a few research on this problem in cloud. The multi-objective character of the scheduling problem in cloud environments makes it difficult to solve, especially in the case of complex tasks [11]. The major research contributions in this field are summarized below.

A load balancing task scheduling algorithm based on weighted random and feedback mechanisms is proposed by

978-1-5386-7503-8/18/$31.00 ©2018 IEEE
DOI 10.1109/FiCloud.2018.00018

Qian et al. [12]. In the first stage of the algorithm, the chosen cloud scheduling host selects resources by demands and sort them based on static quantification. Secondly, the algorithm randomly choses resources from sorted list based on their weight; then it obtains corresponding dynamic information to make load filter and sort the remaining. At last, it reaches, in a self-adaptively manner, the right system load through feedback mechanisms. The experimental results show that the algorithm avoids the system bottleneck effectively and achieves balanced load as well as self-adaptability to it.

Malik et al. [5] developed an efficient priority based round robin load balancing technique which prioritizes various tasks to virtual machines on the basis of various metrics, such as required resources or processors, number of users, time to run, job type, user type, software used and cost. Then, it conveys them to various available hosts in round robin fashion. This approach seems to improve the system capability by enhancing various parameters such as fault tolerance, scalability and overhead, and by minimizing resource utilization and response time. This technique has been simulated and tested over CloudSim, which is a widely used tool to test cloud based techniques.

In another research [13], a hybrid ACHBDF (Ant Colony, Honey Bee with Dynamic Feedback) load balancing method for optimum resource utilization in cloud computing is presented. The developed ACHBDF method uses combined strategy of two dynamic scheduling methods with a dynamic time step feedback method. The proposed ACHBDF utilizes the quality of ant colony method and honey bee method in efficient task scheduling. The used feedback strategy checks system load after each phenomenon in a dynamic feedback table to help migrating tasks more efficiently in less time. An experimental analysis comparing existing ant colony optimization, honey bee method and ACHBDF, showed superiority of ACHBDF.

III. Proposed Method

In this section, a parallel genetic algorithm based approach for task scheduling in the cloud is described. To improve GA, a novel restriction table is used by the proposed algorithm to provide a mechanism to avoid already-checked invalid answers, thus gaining speed while focusing only on validated or potentially new answers. A detailed description of the main components of the proposed genetic algorithm is provided below

A. Chromosome

Genetic algorithm starts with a population of strings (represented by chromosomes), which encodes candidate solutions, called individuals. Each chromosome is represented by an array with the length of *i*, where *i* is the number of tasks to be fulfilled. Each array element, called gene, represents relation of a task and a resource. Each G_i is an integer between 0 and a number of available resources. For example, if G_3 is 8, it means 3rd task will be processed with resource number 8. This way, a chromosome consisting of *N* genes represents a solution (task schedule) for fulfilling *N* tasks with a known number of resources.

G_1	G_2	…	G_{n-1}	G_n

Fig 1. Chromosome sample

B. Initial Population

The initial population defines the speed and the convergence of the genetic algorithm [14]. At the beginning, current population has no member, so *cur_pop = null.* Then, new individuals are created randomly and added to *cur_pop* after they pass precedence order violation test. Every generation of the genetic algorithm must have *N* chromosomes at its start. We use a random method to initialize the population with 100 chromosomes (*pop_size*), which satisfy problem criteria. In each iteration, the best individuals are selected and the worst ones are replaced with new generated ones. To improve random chromosome creation, a list of invalid chromosomes (*inval_chromo*) is stored in a table implemented by an array. When a chromosome is created, it is checked to see if it can be a valid answer. Since a major number of chromosomes can be invalid, it is important to avoid recreating and rechecking those invalid ones that were tested before and failed. This process is both costly and time consuming. So, a list of failed ones is kept separately so that newly created chromosomes are kept apart from that list. The main reason for creating invalid chromosomes of initial population is that any resource could be randomly assigned to fulfill any task, but as each task requests special resource, when algorithm checks validity of a chromosome, it usually fails with a high chance. To avoid this high failure chance, we use *assignable_res,* a useful data structure, that contains actual resources with capability of processing a given task. When the algorithm tries to assign a resource to fulfill a task, it selects an assignable one, so that during check process it won't fail simply because of *req_resType* mismatch.

C. Selection Operator

Roulette wheel selection is a common technique used in GA implementations to select the chromosome. Existing mechanisms select one of *N* individuals using search algorithms that are *O (N)* or *O (log N)* time complexity. In this work, we use a selection algorithm which typically has *O (1)* time complexity, and is based on stochastic acceptance instead of searching. The first step in the selection process is to run all of the chromosomes from the initial population through the schedule builder. After all of the chromosomes have been scheduled and scored, the sum of all of the individual fitness's is calculated, which represents the total fitness for the population.

D. Crossover Operator

The crossover operator combines more than one chromosome to generate the chromosomes of the new generation. The new chromosome inherits some features

from first parent and the remaining features from the other parent. We first define P_C (probability of crossover), and then, the crossover operation starts. Also, a crossover point, CP, is defined to divide a chromosome into right and left sections. CP is a random integer number between 1 and the total number of tasks. The Pseudo code of the crossover operation is described by the following steps:

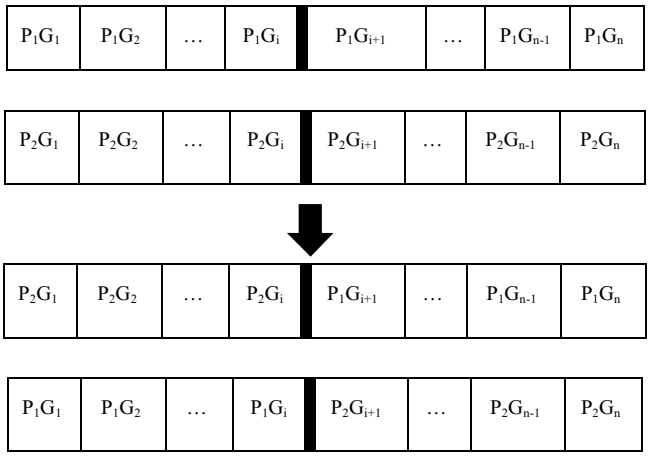

Fig 2. Crossover sample

1. Define P_C in [0, 1]; set i = j = k = 0;

2. Generate a random number r_1 in [0, 1];

3. i++; if (r_1 < P_C) chromosome i is a parent (first one); otherwise proceed from 2;

4. Generate a random number r_2 in [0, 1];

5. j++; If (r_2 < P_C) chromosome i is a parent (second one); otherwise proceed from 4;

6. Define integer CP in [0, N];

7. Divide each parent chromosome into two sections from CP;

8. Swap right side of first parent with left side of second parent;

9. if swapped chromosomes satisfy the problem, save them; otherwise add to inval_chromo; k++;

10. Set i = j = 0; Proceed from step 2 if (k < pop_size);

E. Mutation Operator

In GA, mutation operator is used to maintain the diversity of the population by changing chromosome with a small probability from the interval [0, 1], which is known as the probability of mutation P_M. Also, two mutation points MP_1 and MP_2 are defined to perform mutation operation. These mutation points are random integers between 1 and the total number of tasks. In this work, Pseudo code for the mutation operation is described by the following steps:

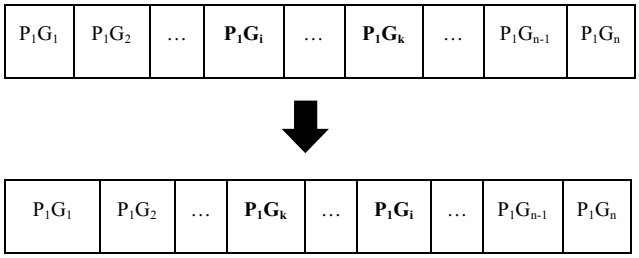

Fig 3. Mutation sample

1. Define PM in [0, 1]; set k = 0;

2. Generate a random number r in [0, 1];

3. If (r < PM) chromosome i is a parent; otherwise go to 7;

4. Define integer MP1 and MP2 in [0, N];

5. Swap two rows (row MP1 and row MP2) of the selected parent chromosome;

6. If the new chromosome satisfies the problem, then save it; otherwise add to inval_chromo; k++;

7. Proceed from step 2 if (k < pop_size);

F. Fitness Function

A service-level agreement (SLA) is defined as an official commitment that prevails between a service provider and a client. SLAs span across the cloud and are offered by service providers as a service-based agreement rather than a customer-based agreement. The decline of cloud computing relative to SLAs is the difficulty in determining the root cause of service interruptions due to the complex nature of the environment.

Arrival tasks are inserted into a waiting queue to be processed. Some tasks might have a deadline to be fulfilled or otherwise they are missed. It is a scheduler priority to lower the number of missed tasks in order to improve overall performance of the system.

An important goal of proposed method is to balance different loads while scheduling tasks. Loads should be distributed over resources so that they have similar traffic. If a resource gets busy while another one is free, tasks with deadline might get failed as they wait to be processed in the busy resource. So, a statistical metric such as standard deviation is needed to measure variation or dispersion of loads. Lower SD shows higher load balance.

Fitness of a chromosome is defined as the minimum value for SLA interruptions, lower missed tasks (MT) and lower standard deviation (SD). Fitness is defined to evaluate efficiency of a built chromosome as a solution. Also, importance of these three factors can be defined with coefficient α, β and γ as shown below.

$$FF = Min\ ((\alpha*SLA)*(\beta*MT)*(\gamma*SD)) \quad (1)$$

For example, if load balancing is more important than SLA and MT, α=0.25, β=0.25 and γ=0.5 seems satisfying. Due to the nature of independency between each chromosome, calculation of fitness value during a generation, which is a time consuming calculation, is done in parallel. For example, in a generation of 100 chromosomes, calculating fitness value of each individual can be done separately in parallel. Also, after crossover and mutation operations, newly constructed individuals' fitness should be measured. If just created chromosome is already in *inval_chromo*, it will be simply ignored as if it was not created. Otherwise it will be checked to be a valid chromosome and its fitness can be calculated in parallel.

G. Termination Condition and Pseudo Code

The algorithm is terminated if no further improvement in the fitness value of the best chromosome in the population is not occurred for ten iterations or maximum number of generation is reached. The Pseudo code of the algorithm is described by the following steps:

1. *Initialize pop_size chromosomes randomly;*
2. *Calculate the fitness of each chromosome in parallel;*
3. *Perform crossover in parallel;*
4. *Perform mutation in parallel;*
5. *Evaluate the fitness of the offspring in parallel;*
6. *Select the survived individuals;*
7. *Proceed from step 3 if termination criteria have not been fulfilled.*

It is obvious that most repeated operations are being done in parallel. This way, time consuming actions are controlled and a vivid disadvantage of GA can be healed.

IV. Simulation

The proposed method is simulated using Matlab and compared with two similar methods using various scenarios. Running environment includes Matlab 2016b on a 64-bit system with an Intel Core i7-6700 processor and 32GB DDR3 RAM. Table I summarizes the important parameters used during simulation and for comparison of the proposed method with the hybrid method [13] & RR method [5], described in the related works section.

TABLE I. Important Parameters Used in Scenarios

Size of population	100
Maximum number of generations	1000
Selection mechanism	Roulette wheel via stochastic acceptance
Fitness function	as defined in equation (1)
Crossover rate	One point (P_C = 0.9)
Mutation rate	Swap (P_M=0.05)
Termination conditions	No improvement in the fitness value of the best chromosome for ten iterations.
Size of population	100

The proposed method is compared with the two above-mentioned methods which are simulated in Matlab too. The dataset for fair and standard comparison is CoMon [15]. CoMon actively gathers a number of metrics useful for developers of networked systems, collects and reports statistics on roughly 400-450 active PlanetLab nodes, and 200-250 active experiments running on PlanetLab. Workload of year 2011 is used as sample dataset which contains various situations where some tasks and resources with different conditions exist in the system. Distinct scheduling methods get these properties and try to reschedule and manage these situations to gain more efficient answers. Each method has its unique solution which demands some changes in order to complete given tasks. So, some tasks might be assigned to different resources, which requires migration. Migration is a costly action in cloud, which reduces overall performance of the system. So, lower number of migration means better balance among resources as they work in a stable condition.

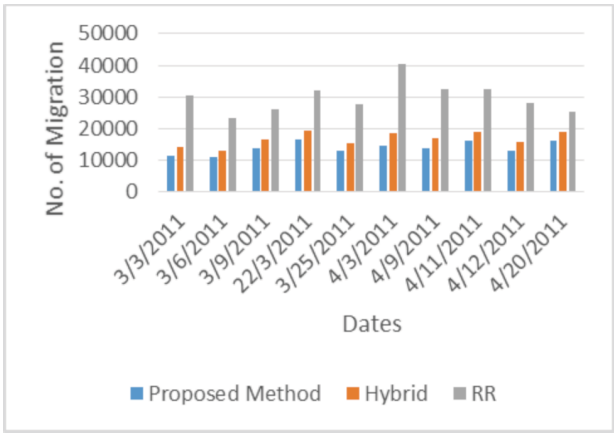

Fig 4. Comparison of migration

Fig. 4 shows the number of migrations involved in the three methods. These results demonstrate that the proposed method demands lower motion among resources between 14% - 37%. As capable resources are assigned to each task (outcome of restriction table assignment), migration rate is low and load balancer algorithms manage ongoing loads on each resource easily without interruption. On the other hand, higher migration means difficulty in prediction of loads, which demands tight monitoring of each resource to keep them balanced again. So, the results show stability in system which confronts disturbing load balance in task fulfilling.

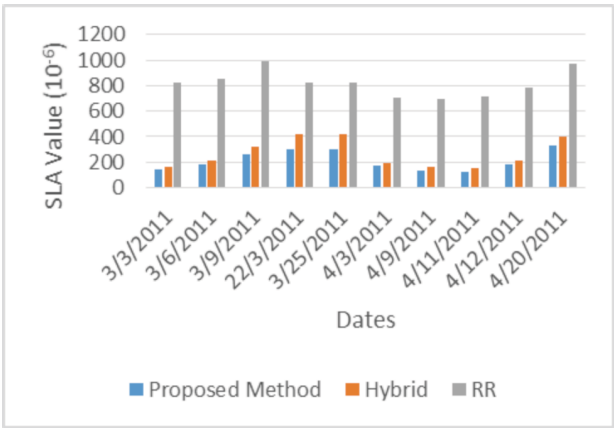

Fig 5. Comparison of SLA interruptions

In fig 5, a comparison of 10 days SLA violations is illustrated. It is obvious from the observed results that the proposed method is acting well in comparison with other two methods in SLA interruptions. Overall system performance calculation shows a 13%- 17% improvement in comparison with the two other algorithms. SLA interruption rate is inversely related to system efficiency. Lower interruptions resulted from a stable condition among resources, where suitable resources are carefully selected to fulfill tasks and loads are balanced. Also, when SLA quality improves, user satisfaction grows mutually.

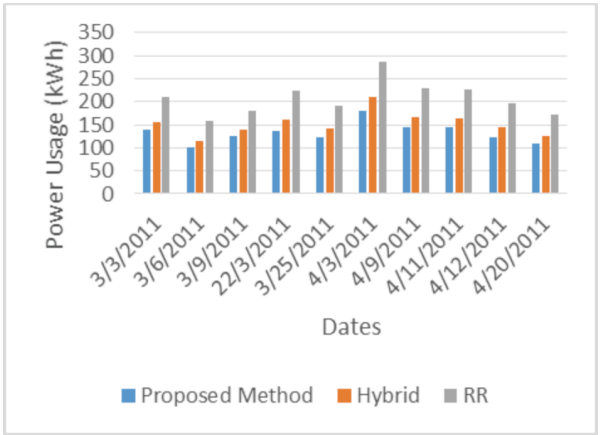

Fig 6. Comparison of power usage

Finally, another comparison is shown in fig 6. It illustrates that the proposed method demands lower power, estimated in this study between 9% - 31%. When loads of resources are balanced, overall power usage decreases. This is due to automatic monitoring of resources so that they go to stand-by mode if they are not busy. So, higher load balance results in lower energy usage of the overall system.

V. Conclusion & Future Work

In this paper, we proposed an algorithm for task scheduling in the cloud. The proposed method is a parallel genetic algorithm-based method which schedules tasks with priorities. It aims to improve load balancing rate while better resources are selected to fulfill arrival tasks in a shorter time with lower task fail rate. The method is simulated in Matlab and compared with two similar methods, hybrid and RR methods, using various scenarios. The obtained results show that the proposed method has 9% - 31% lower energy usage, 14% - 37% lower migration rate, and 13%- 17% lower SLA violation, thus proving its robustness as an efficient alternative load balancer. As it is an NP-Complete to schedule tasks, other new meta-heuristic methods could be found to further improve this vital problem which requires lower run time.

References

[1] Navimipour, N.J., et al., Expert Cloud: A Cloud-based framework to share the knowledge and skills of human resources. Computers in Human Behavior, 2015. 46: p. 57-74.

[2] Buyya, R., et al., Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, 2009. 25(6): p. 599-616.

[3] Reddy, J.G., A Review Work On Task Scheduling In Cloud Computing Using Genetic Algorithm. International Journal of Technology Enhancements and Emerging Engineering Research, 2013. 2(8): p. 241-245.

[4] Maurer, M., I. Brandic, and R. Sakellariou, Adaptive resource configuration for cloud infrastructure management. Future Generation Computer Systems, 2013. 29(2): p. 472-487.

[5] Malik, A. and P. Chandra, Priority based Round Robin Task Scheduling Algorithm for Load Balancing in Cloud Computing. Journal of Network Communications and Emerging Technologies (JNCET) www. jncet. org, 2017. 7(12).

[6] Khezr, S. N. and N. J. Navimipour (2017). "MapReduce and Its Applications, Challenges, and Architecture: a Comprehensive Review and Directions for Future Research." Journal of Grid Computing 15(3): 295-321.

[7] Ashouraie, M. and N. Jafari Navimipour, Priority-based task scheduling on heterogeneous resources in the Expert Cloud. Kybernetes, 2015. 44(10): p. 1455-1471.

[8] Xu, Y., et al., A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Information Sciences, 2014. 270: p. 255-287.

[9] Katyal, M. and A. Mishra, A comparative study of load balancing algorithms in cloud computing environment. arXiv preprint arXiv:1403.6918, 2014.

[10] Tao, F., et al., CLPS-GA: A case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. Applied Soft Computing, 2014. 19: p. 264-279.

[11] Abrishami, S. and M. Naghibzadeh, Deadline-constrained workflow scheduling in software as a service cloud. Scientia Iranica, 2012. 19(3): p. 680-689.

[12] Qian, Z., et al., A load balancing task scheduling algorithm based on feedback mechanism for cloud computing. International Journal of Grid and Distributed Computing, 2016. 9(4): p. 41-52.

[13] Pawar, N., U.K. Lilhore, and N. Agrawal, A Hybrid ACHBDF Load Balancing Method for Optimum Resource Utilization In Cloud Computing. 2017.

[14] Abdoun, O., J. Abouchabaka, and C. Tajani, Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem. arXiv preprint arXiv:1203.3099, 2012.

[15] Park, K. and V.S. Pai, CoMon: a mostly-scalable monitoring system for PlanetLab. ACM SIGOPS Operating Systems Review, 2006. 40(1): p. 65-74.