International Journal of Control and Automation
Vol. 9, No. 6 (2016), pp.105-118
http//dx.doi.org/10.14257/ijca.2016.9.6.10

Load Balancing in Decentralized Grid Scheduling Systems Using Bee Colony Algorithm

Asgarali Bouyer

Department of information technology, Azarbaijan Shahid Madani University, Tabriz, Iran
a.bouyer@azaruniv.edu

Abstract

Load balancing is an important part of grid scheduling systems which aims to utilize dynamic and decentralized grid resources without any overcapacity happening at any resources. Load balancing is being considered as an important phase for distributed computing environments. In large scale distributed systems such as grid/cloud computing, it is necessary to use a distributed load balancing system in job scheduling system. Distributed load balancing was identified as a major concern to allow grid computing to scale up the job scheduling. Many algorithms had been proposed for finding the solution of load balancing problem in these fields. But very few algorithms are proposed for distributed load balancing in grid computing environments. In this paper, we have developed an adaptable artificial bee colony (ABC) algorithm for dynamic dividing and scheduling jobs via decentralized schedulers. The proposed method significantly enhances the load balance on the resources. Besides, this method does not need primary information about the capacity and power of the resources which is compatible with the variable and heterogeneous nature of decentralized scheduling. The proposed method is compared with the cuckoo based scheduling algorithm and OSL decentralized scheduling method. The results of the simulations revealed that the proposed method is significantly better than the OSL method. It also outperforms the cuckoo based scheduling algorithm in most of simulations.

Keywords*: Load balancing, Job scheduling, bee colony algorithm, schedulers' communication, decentralized scheduling*

1. Introduction

Grid computing is aggregation of autonomous resources that are geographically distributed. The nodes in grid permit sharing and selection dynamically at runtime [1]. These distributed computational resources are perceived by the user or applications as a virtual vast computational system. With the emergence of such computational systems, parallel and distributed computations and also application programs which formerly required supercomputers can now be easily executed. This technology is mainly used in applications which need high processing. Thus, it can be argued that the main idea behind grid is the sharing of the processing capacity and ability of the resources which are geographically scattered and distributed and can communicate with each other through fast networks. Since these resources belong to different people and organizations, grid realizes resource sharing and problem solving in dynamic virtual organizations. In grid, virtual dynamic organizations are linked with each other through middleware and provide basic service layers such as resource management, supervision and security [2].

Dynamic resource allocation is required in grid environment because of dynamic resources and requests [3]. This essential dynamism in the grid computing requires well-organized load balancing mechanisms. Load balance algorithms in the central processing unit are costly and they need a lot of time for processing which result in high complexity

ISSN: 2005-4297 IJCA

and non-homogeneity. Inasmuch as the number of resources in grid environment is high, maintaining load balance between jobs and resources is difficult. Like in, all other internet based distributed computing tasks, load balancing is an important aspect in cloud computing [4].

Many researchers have proposed different procedures for solving this problem. The proposed solutions have tried to reduce the number of idle resources in grid so that all the resources are active rather than having a few active resources and a few inactive resources. This is considered to be the responsibility of schedulers. As this component operates better, a better load balance will be achieved. With respect to the number of available schedulers in grid, it can be divided into two general categories: 1) in the first category, there are grids which have only one centralized scheduler; such grids can optimally maintain load balance between resources. However, with respect to the heterogeneity of resources, the number of resources and requests, the use of a scheduler is not ideal and desirable due to the presence of heavy load on it and heavy network traffic [5]. In the second category, for solving the problems caused by using one scheduler, a few (decentralized) schedulers rather than one scheduler are used. In this category, the schedulers have been distributed through the entire environment. Each scheduler receives tasks from a few users and allocates resources to them. The users send tasks to the scheduler which is geographically closer to them. Hence, the amount of loads in the schedulers and in the grid decreases [5]. However, it should be mentioned that the presence of several schedulers which operate independently of one another causes a different problem. That is, maintaining the optimal load balance on resources is a challenge in itself. The first solution for sorting out this issue is that the resources declare the amount of tasks to be done to the schedulers. In other words, schedulers should be aware of the amount of tasks accumulated on each resource [6]. Nevertheless, this approach not only imposes further load on the grid but also uses the processing power of the resources for obtaining report and sending it to the schedulers. In another approach, schedulers use an indirect method to Figure out the amount of load on resource. In this approach, two methods can be used for establishing communication among schedulers and obtaining statistical data about resources. In the first method, the best condition is that all the schedulers are connected to a grid information service (GIS) and they update their data each time they allocate resources to the tasks. However, such a system can have the same challenges mentioned for the centralized grid. That is, schedulers are required to communicate with the GIS twice (once for obtaining data about resources status and a second time for updating data) to schedule a task [7]. Another approach is to utilize local GIS and linking them with higher level GIS. In this case, the tree structure can have more levels. Hence, each GIS will deal with a more limited number of schedulers. As a result, GIS loads will be significantly reduced. Nevertheless, coordinating GIS is regarded as a complex issue in the management of several local GIS. One more approach for establishing communication among schedulers is to link them to each other without using GIS [5]. In this approach, each scheduler is linked to another scheduler and shares its data with those of other schedulers. In this way, the schedulers are organized with respect to a certain parameter such as geographical distance from each other. As a scheduler receives a task, it gets connected to the scheduler to that and combines its data with its own data; then, based on the new status, it allocates a resource to the task. Not unlike the above-mentioned approaches, the problem of additional load exists in this approach. Moreover, sharing data between two schedulers might result in the production of invalid and unreliable data with respect to the resources.

Inasmuch as several challenges and problems are involved in establishing communication among schedulers which were shortly overviewed above, in recent years, researchers have attempted to ignore the establishment of communication among schedulers. Rather, researchers in this field have tried to indirectly obtain data about resources. According to this recent approach, schedulers are unaware of each other.

However, based on resource delay in fulfilling the allocated tasks, the schedulers obtain data about the amount of accumulated tasks in each resource.

Figure 1 illustrates the outline of grid environment division which has been proposed by the present paper. The present paper has proposed a novel method which combines the obtained statistical data and in this way, it is informed about the status of the resources. In the proposed method, the artificial bee colony algorithm is used to schedule tasks in the decentralized method in the grid so that load balance on resources is optimized. In the method put forth in this paper, the schedulers have no communication with each other and there is no need to mention the amount of resource capacity to the schedulers. Therefore, the amount of network traffic will be minimized. The proposed method is compatible with completely dynamic environment where resource capacity is likely to change at any moment.

The reminder of the paper is organized as follows: section two will review the previous works. Then, section three will mention the hypotheses of the present study. Section four will describe the methodology of the study and data collection procedure in detail. In section five, the method of using artificial bee colony algorithm will be extrapolated. In section six, the results of the experiments will be given and the proposed method will be compared with cuckoo based scheduling algorithm and OSL method. Finally, section seven will summarize the study, pinpoint the findings of the study, suggest further directions for research and draw the conclusion of the study.

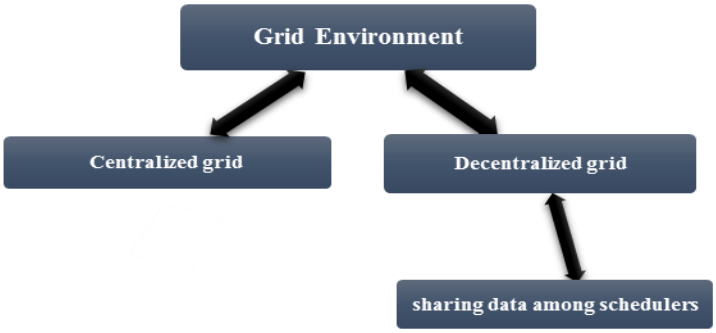

Figure 1. Grid Environment Division Based On the Number of Schedulers and the Communication between Them

2. Research concepts and Previous Works

Load balancing policy is vital in distributed computing systems (*e.g.* grid and cloud computing) due to stability the load of resources with aims to maximize resource utilization and performance. This policy is complicated with grid computing systems. Grid environments possibly has a massive number of machines. The state of the grid dynamically changes and the load balancing policy should adapt its decisions according to new status [8]. Load balance mechanisms can be divided into two categories: the first category incorporates centralized and decentralized load balance, the second category includes static and dynamic load balance [9]. In a centralized approach, all scheduling decision are made at one node or site and the failure in central node cause entire system down. in other words, in centralized scheduling, the grid scheduler had total control over resources. However, this method is not appropriate and scalable for a highly huge environment such as grid. A method proposed based on centralized load balance is a kind of genetic algorithm in which the central scheduler uses a coding process for genes. Under this process, a group of coded genes is produced which are known as chromosomes; chromosomes are manipulated by genetic operators so that final chromosomes are

obtained. For proper manipulation of chromosomes, a qualification function is required so as to control genetic operators and measure the efficiency of chromosomes [10, 11]. This algorithms assumes that all the jobs in the system are scheduled by a centralized grid scheduler.

In a decentralized algorithm, there are not any nodes as information collectors or service providers. Rather, all nodes have data and information about some or all of the nodes; this results in a great communication load. It should be noted that this information is not reliable because data should be updated [10, 11]. One of the proposed decentralized algorithms is one which is based on an intelligent ant colony. In such an algorithm, there is an ecosystem of intelligent ants. The interaction among independent intelligent ants results in load balance all over the grid. The ants record and register the characteristics of the nodes which they meet in their hops so that they can use that information in their future decision making. Then, the ants exchange these information with each other so that the number of overload and under load nodes become equal. In other words, load balance in this algorithm is maintained by means of ants and these ants operate in harmony with the environment. Indeed, ants can move from one node to another node and in case one node has overload, ants can balance the load of that node with those of other nodes [12]. Decentralized algorithms optimizing decentralized goals include mainly economic approaches [13]. Decentralized algorithms optimizing system-level goal, that proposes a load-balancing algorithm for divisible task model [14].

In a decentralized algorithm, some approaches is made scheduling and load balancing decision in a static manner. Static load balance makes use of data such as operation mean, operation period, *etc*. and based on these data, mathematical formula are used to transform tasks into a different method so that each node can fulfill the task which was allocated to it [15]. The advantage of this method is that system information need not be gathered at each moment so that they can be updated at any moment. However, it should be pointed out that the efficiency rate of some nodes is low. That is, they might be used less than other nodes and hence, they do not proceed dynamically according to system information [16]. Another algorithm which has been proposed based on this approach uses the following four methods: random method, round robin, threshold and central management algorithm. The response time of each method was evaluated; based on the evaluations of response time, the central management has the least response time among the four methods [17]. Furthermore, in our previous research, an Adaptable Job Submission System has been proposed for Market-Based Grids in which the load balancing is considered as a criterion for selecting suitable resources [18].

In dynamic load balancing the scheduling decisions are made when there is need to schedule the job for next processing. A dynamic task scheduling can use either centralized or distributed control. Dynamic load balance operates based on the current status of the system; that is, at any given moment, it determines that which task should be allocated to which node. In case there is overload in a node, the task will be allocated to another node and it will be processed by that mode. However, it should be mentioned that task migration from one node to another node results in an overload [16]. An algorithm which has been proposed based on this approach includes a branch and bound algorithm in order to maintain load balance. According to this approach, a node weight and a node are defined for each of the nodes. Then, based on this information, an interval is defined; these intervals are determined based on specific operators. Load balance in this method is operationalized by certain processors and a coordinator. Using this method results in cost optimization [19]. Mehta *et al*., proposed a decentralized content aware load balancing approach for distributed computing environments. This load balancing strategy is implemented in a decentralized manner with low overhead [20]. cuckoo based scheduling algorithm[21] is another algorithm that dynamically attempts to find optimal solution for job scheduling using cuckoo optimization algorithm. In this paper, we have compared our proposed method with this method.

3. Honey Bee Behavior Inspired Load Balancing Method

In the method which has been proposed in this paper, the researchers will fist mention the related hypotheses in the grid environment. Then, the statistical data which was used in this study to evaluate grid resources will be described. It will be mentioned that how these statistical data were gathered. Next, the proposed artificial bee colony algorithm will be described in detail. In this section of the paper, it will be mentioned that how the gathered data will be used to allocate resources to the tasks.

3.1. Assumptions of the Study

By default, in new methods, a certain amount of abstraction is used for scheduling tasks in grid environment. The purpose of using abstraction is to eliminate the complexities which have no effect on solving the problem by the new method. Likewise, in the present study, to simplify the grid environment, enhance the concentration on solving the scheduling problem and maintain load balance, the researchers have assumed some assumptions about the grid environment. However, it should be noted that after eliminating these assumptions, the proposed method is still applicable in the grid environment. The assumptions include the followings:

1. The number of schedulers in the grid environment is limited which is denoted by N.
2. Each scheduler receives tasks from a number of users. The number of received tasks by each scheduler can be denoted by the λ rate and Poisson distribution. The magnitude of a task is denoted by an integer.
3. Each resource is a process with a specific speed which is expressed by the amount of realized task at the unit of time.
4. The number of resources in the grid environment is limited which is denoted by M.
5. The time delay in the communication among resources and schedulers has been disregarded and ignored.

3.2. The Statistical Data Used in the Proposed Method

As it was mentioned earlier in the introduction of the paper, data about resources should be obtained so that in the next stages, the gathered data can be used for allocating tasks to the resources. In general, in grid computing systems, there are two types of obtaining statistical data:

- *Static gathering of statistical data*: in this kind of statistical data collection, at the outset, static data are provided for schedulers. These data include resource characteristics such as speed and capacity of resources. The problem with this method is that, due to the dynamic nature of grid, resource capacity might change based on the allocated tasks. Hence, scheduler data will be unreliable. To sort out this problem, an attempt has been made to obtain data at different and alternative times which is referred to as dynamic collection and gathering of data.
- *Dynamic data collection*: in this method, based on the tasks which scheduler has allocated to each resource and with respect to the response time of the resources, the scheduler obtains information about the capacity of each resource. Information about the resource capacity can be obtained either at the time of allocating a task to each resource or at different time intervals.

In the method proposed in this paper, data about resource capacity is obtained in the following way: at the outset, it is assumed that the schedulers have no information and data about the capacity of resources. Each scheduler has two vectors, namely *weight* and *utility* with the length of M (the number of resources). In *weight*vector, the total amount of tasks allocatedby scheduler to each resource is

stored and in the *utility* vector, the capacity of each resource is stored. The initial value of weight vector is zero. Any time a task is sent to a resource, the task length is summed with the corresponding element (value) in weight vector. When that task is finished, its amount is subtracted from the weight vector. The initial value of utility vector is zero and it is obtained through equation (1) below:

$$Utility_{res(j)} = \frac{|w_i|}{t_2 - t_1} \qquad for\ \mathrm{j} = 1, 2, \ldots \mathrm{M} \tag{1}$$

Where res(j) refers to the j[th] resource, |Wi| denotes the magnitude and amount of the i[th] task which has been allocated to this resource. The time t_2 denotes the required time for the completion of the task; and t_1 refers to the required time for transmitting a task to the resource. The above-mentioned equation indirectly provides the scheduler with the capacity of each resource. Based on the values of weight and utility vectors and using the artificial bee colony algorithm, the scheduler makes the required decisions for the next stages (indeed, these two vectors provides the value which artificial bee colony algorithm needs to measure the qualification of each resource).The amount of qualification of each resource is obtained through equation (2) below:

$$fitness_{res(j)} = \frac{Utility_j}{weight(j) + 1} \tag{2}$$

Where $fitness_{(res(j))}$ refers to the amount of qualification of j^{th} resource. The rationale for using weight vector in the measurement of the degree of resource qualification is that the amount of sent tasks to a certain resource should not exceed a specific limit. In other words, if only utility vector is used to determine the degree of qualification of resources, the task load of high-capacity resources will be extremely high. Hence, the overall efficiency of the system will be low. It should be noted that equation (2) causes a reduction in the efficiency of a resource when a task is allocated to that resource and the next times, the same resource might not be selected as a top resource.

3.3. Artificial Bee Colony in Task Scheduling

In the preceding section, a method was mentioned for determine the qualification amount of resources. However, there is a need for an appropriate algorithm which can use these qualification values. In this paper, the researchers have proposed the artificial bee colony algorithm. The artificial production process used by artificial bees in the nature is one of the intelligent behaviors which has inspired researchers in the area of artificial intelligence. Indeed, researchers have made use of this process in optimization issues [22].In the colony of artificial bees, there are three types of artificial bees: Scout bees, Follower bees, and Dancing bees. The scout bees' task is to search for flowers with high nectar in the surrounding environment. These scout bees inform the dancing bees of their search result. The dancing bees perform dancing vibrations and movements based on the quality of the search results. That is, the better is the search results, the longer will be their dance. At this time, the follower bees notice the dancing bees, select one of them and follow them towards the target which they guide. It is clear that the attraction of more follower bees is a function of the length of their dance.

Likewise, in this paper, the same intelligent algorithm was used to allocate tasks to resources. However, some minor alternations were made to the initial artificial bee colony algorithm. In the algorithm proposed in this paper, the tasks were assumed to be the artificial bees, the schedulers were regarded as hive and the resources were deemed to be the flowers. In general, the proposed algorithm

included two stages: the first stage begins when the scheduler starts to work and it will continue as long as we have no information about the resources. The second stage begins when the initial information about the schedulers is obtained and it continues until the scheduler task is finished. In order to determine the artificial bee type, we have used the following procedure: since we have no information about the capacity and resource load (flowers) in the first stage, all the tasks are assumed as the scout bee. Then, the scout bees are randomly allocated to the resources (flowers). This stage is continuously done until the scout bees come back to the scheduler (hive) and give dancing bees some information about the resources. In general, scout bees return to the scheduler (hive) when the tasks are completed by the resources. In other words, bees' coming back to the hive refers to the completion of the task by the resource and taking the results back to the scheduler. At this time, in harmony with the resource, the dancing bee uses the equations (1) and (2) to measure and determine the qualification degree of the respective resource. It should be mentioned that in each scheduler (hive), there is one dancing bee for each resource whose task is to advertise its own resource. The dancing bees not only determine and update the qualification degree of their own resources at the end of a task but also determine and update the qualification of their own resources at the allocation time of tasks; this is because the house corresponding with that resource in weight vector has changed. In the second stage, a task which has just arrived at the scheduler is more likely to be regarded as a follower bee or a scout bee. In general, in the second stage, the new task might be regarded as a follower bee with the probability of γ or as a scout bee with the probability of $(1-\gamma)$. If the task is assumed to be a scout bee, it will be randomly sent to a resource and if the task is assumed to be a follower bee, a resource will be selected which currently has the highest qualification. In both cases, the dancing bee which corresponds with the selected resource will update the qualification of the resource. The following pseudo-code illustrates the proposed algorithm through a formal language and Figure 2 shows the pseudo-code of the proposed algorithm.

```
(1) While (true)
(2)    If there is new job for scheduling.
(3)          If there is no information about resources.
(4)                Take new job as scout bee.
(5)                Assign a resource to scout bee randomly.
(6)          Else
(7)                Due to the probability γ take new job as follower or scout bee.
(8)                If new job is scout
(9)                      Assign a resource to scout bee randomly.
(10)               Else
(11)                     Assign the resource with highest fitness to follower bee.
(12)         End
(13)         Update fitness of resource by eq. 2.
(14)   End
(15)   If there is finished job, update fitness of corresponding resource by eq. 1 and 2.
(16) End
```

*Figure 2. Pseudo-Code of the Proposed Algorithm for Scheduling Tasks

4. Results of the Experiments

In this section of the paper, the proposed method will be evaluated and compared with another method such as cuckoo based scheduling algorithm[21] and OSL [5]. Cuckoo based scheduling algorithm implements cuckoo optimization method for

solving job scheduling problem in the computational grids. In the OSL method which has been proposed for scheduling in decentralized grids, the final decision is made based on two different processes. In the first process, each scheduler keeps the efficiency value for each resource which is zero at the beginning. When a task is sent to a resource, the scheduler subtracts one unit from the efficiency value of that resource; then, if the task is not completed at any given time unit, again, one more unit is subtracted from the efficiency value of the unit.

Another process shares the available tables among the schedulers. Hence, each scheduler receives the efficiency table of the resources of the former scheduler before it sends a task to a resource. Then, it combines the values of the received table with the values of its own table. Therefore, the scheduler obtains further information about the overall status of the resources. In the OSL method, each task entered given to the schedulers is sent to a resource which has the highest efficiency value.

4.1. Simulation and Applied Parameters

The proposed method in this paper is implemented in the Matlab software along with OSL method and cuckoo based scheduling algorithm. The simulation system was carried out on a computer which had the processing frequency of 2.4 GHZ; it had a double-core CPU and a 3 GB RAM memory. The assumptions which were considered about the grid environment were as follows:

The grid environment included N scheduler and M resources where each scheduler was connected to all the resources. Each resource included one processor whose capacity was C; the capacity indicates the number of commands conducted by the processor at the unit of time. The capacity of the resources was within the interval $[C_{min}, C_{max}]$. Each scheduler communicates with a number of users that send tasks to the schedulers. The rate of tasks received by the schedulers depends on the Poisson distribution with the meanλparameter. The magnitude of the tasks is within the range $[J_{min}, J_{max}]$ which indicates the number of available instructions in each task.

In this section of the paper, two different scenarios were taken into consideration for evaluating the proposed method and other algorithms. In the first scenario, the efficiency of the methods was considered in the grid environment and in the second grid, the efficiency of the methods was considered in a large grid environment. The parameters common among these scenarios are given in Table 1 below. It should be noted that ABC refers to the proposed method in all the given Figures.

Table 4. Common Parameters among the Simulation Scenarios

Name of the parameter	Value
$[C_{min} , C_{max}]$	**[50,350]**
$[J_{min} , J_{max}]$	**[5, 995]**
λ	**0.93**
OSL(α)	**0.5**
OSL(β)	**0.5**
The ratio of follower bees to scot bees γ	**0.7**

4.2. Comparing the Proposed Method with OSL

The above-mentioned different scenarios were implemented on the proposed method as well as cuckoo based scheduling algorithm and OSL method; and their results were compared with each other. The simulation results included mean values, maximum and standard deviation of the mean value for the *Makespan* of all the resources in each stage of the simulation. Makespan is one of the standards in

examining the amount of load balance on resources in Grid environment; it reveals the ratio of the total amount of tasks waiting for a resource to the processing capacity of that resource. Equation (3) shows the procedure for obtaining Makespan value for a resource entitled res_j:

$$Makespan_{res(j)} = \frac{\sum_{i=0}^{n} | w_i |}{c} \tag{3}$$

Where $|w_i|$ stands for the magnitude or amount of i^{th} task and n refers to the total number of tasks. Also, c stands for the processing capacity of the resource. If one wants to obtain mean, maximum and standard deviation for the *Makespan* criterion of the resources in each step of the simulation, then, the equations (4), (5) and (6) should be applied on the Makespan values of each of the resources.

$$Makespan_{Ave} = \frac{\sum_{j=0}^{M} Makespan_{res(j)}}{M} \tag{4}$$

$$Makespan_{\max} = Arg\max\{Makespan_{res(j)}\} \tag{5}$$

$$Makespan_{std} = \sqrt{\frac{(Makespan_{Ave} - Makespan_{res(j)})^2}{M}} \tag{6}$$

In these equations, M stands for the number of resources. It should be pointed out that the purpose of all the three equations is the minimization. That is, as the values of mean, maximum and standard deviation are smaller, it will indicate a better performance for a resource. In the first scenario, the number of schedulers was considered to be 30 and the number of resources was 100. Figure 3 illustrates mean Makespan and Figure 4 depicts maximum Makespan while Figure 5 shows the makespan standard deviations for the two methods in comparison with each other in each step of the simulation.

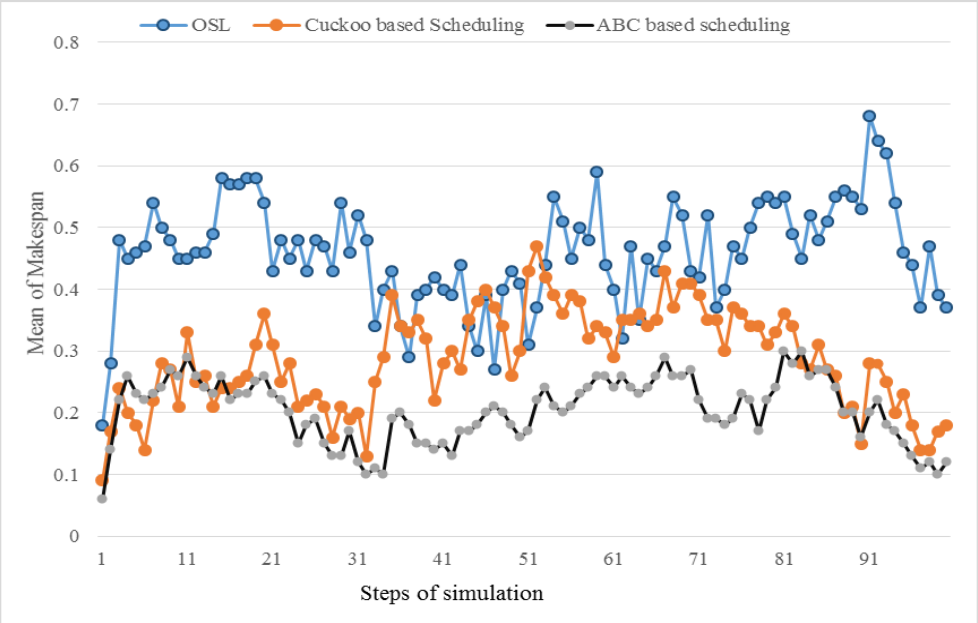

Figure 3. Mean Makespan in Each Step of the Simulation with 100 Resources and 30 Schedulers

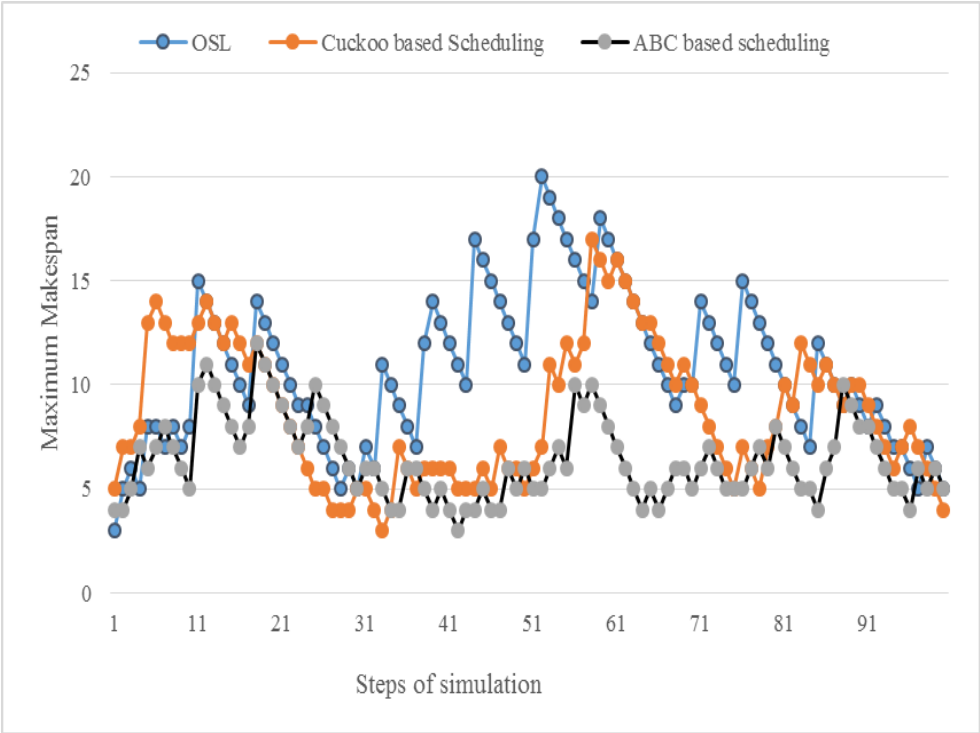

Figure 4. Maximum Makespan in Each Step of the Simulation with 100 Resources and 30 Schedulers

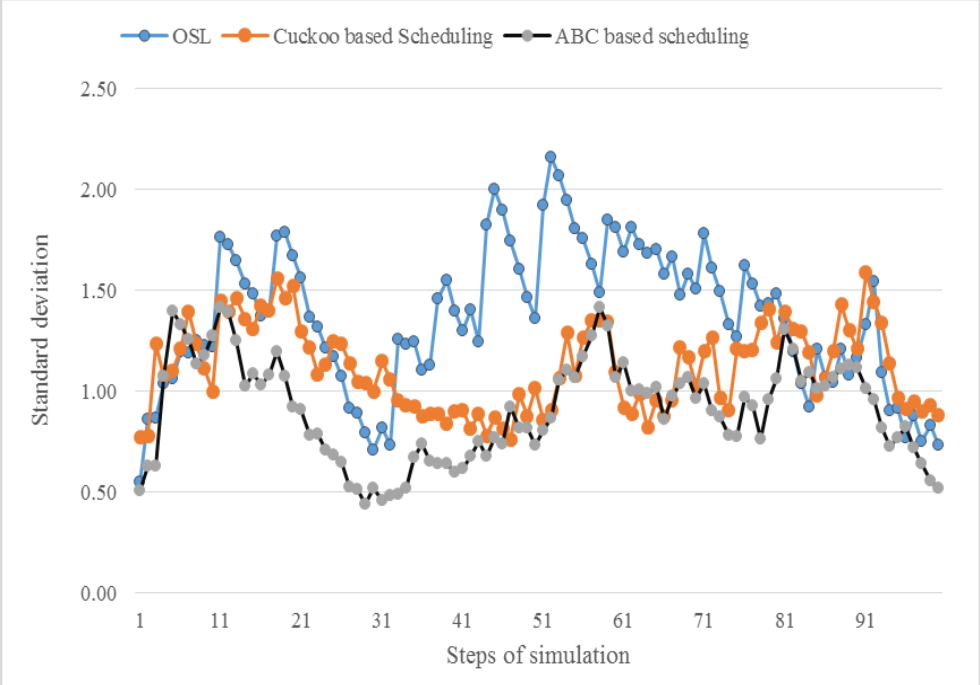

Figure 5. Makespan Standard Deviations in Each Step of the Simulations with 100 Resources and 30 Schedulers

As shown in these Figures, although in the method proposed in this paper, the schedulers do not communicate with each other, it has performed better than cuckoo based scheduling algorithm and OSL in which the schedulers communicate with each other. Noting the above-mentioned Figures reveals that OSL method highly

increases the available load on some resources while the available load on other resources is low. One can Figure out this by noticing the high standard deviation in the initial steps of the simulation. The reason for the load rise in OSL is that schedulers have no information about the resources at the beginning; hence, their sharing of unreliable and invalid information with each other results in worse conditions. However, the conditions change for better as the schedulers gradually obtain more information about the resources. In contrast, the method proposed in this paper does not suffer from this problem since in the proposed method, the resources are randomly selected and this reduces the resource selection interference and overlap of the schedulers. In other words, in the proposed method, the probability of selecting a resource which has been previously selected by another scheduler is extremely low. In the second scenario, 100 schedulers and 250 resources were taken into consideration. Figures (6), (7) and (8) respectively demonstrate mean, maximum and standard deviations for the Makespan criterion of the resources in each step of the simulation for the two methods.

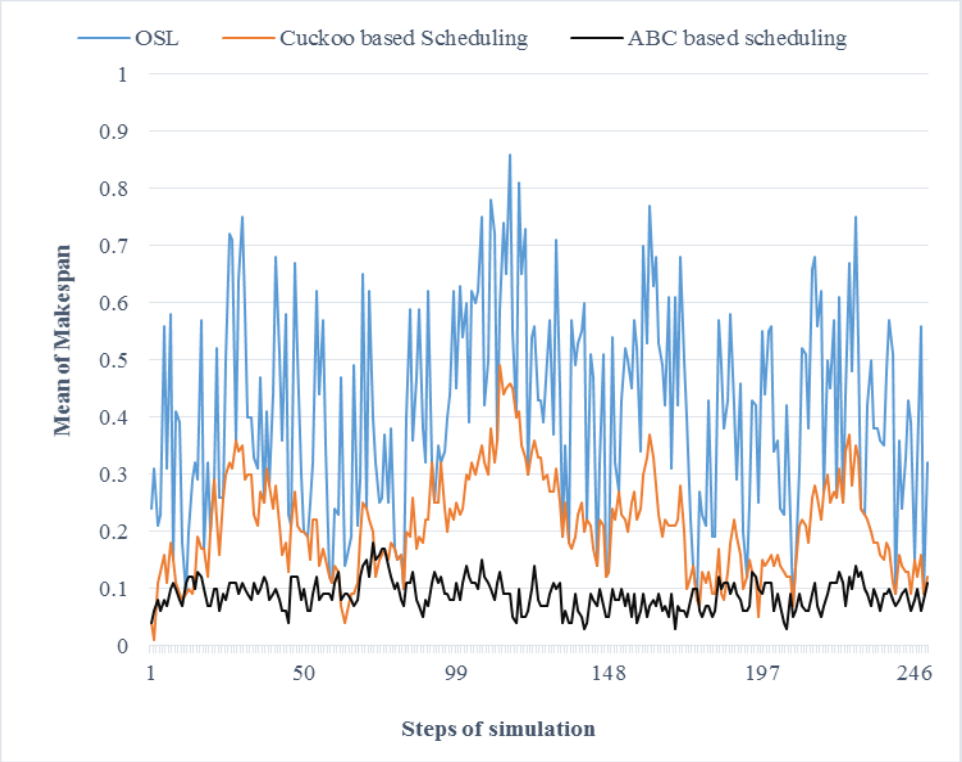

Figure 6. Mean Makespan in Each Step of the Simulation with 250 Resources and 100 Schedulers

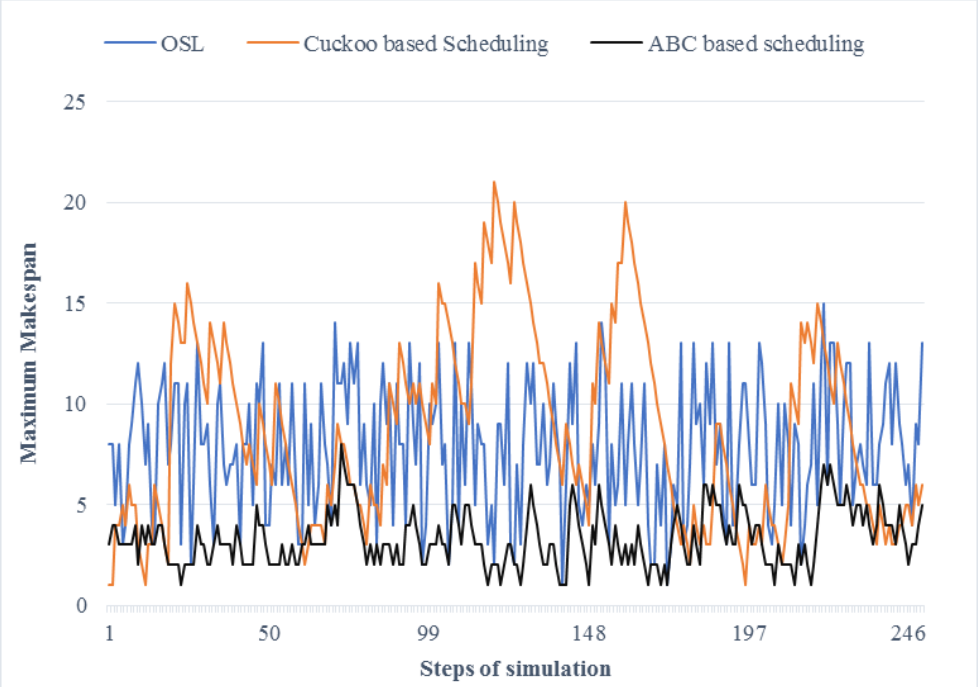

Figure 7. Maximum Makespan in Each Step of the Simulation with 250 Resources and 100 Schedulers

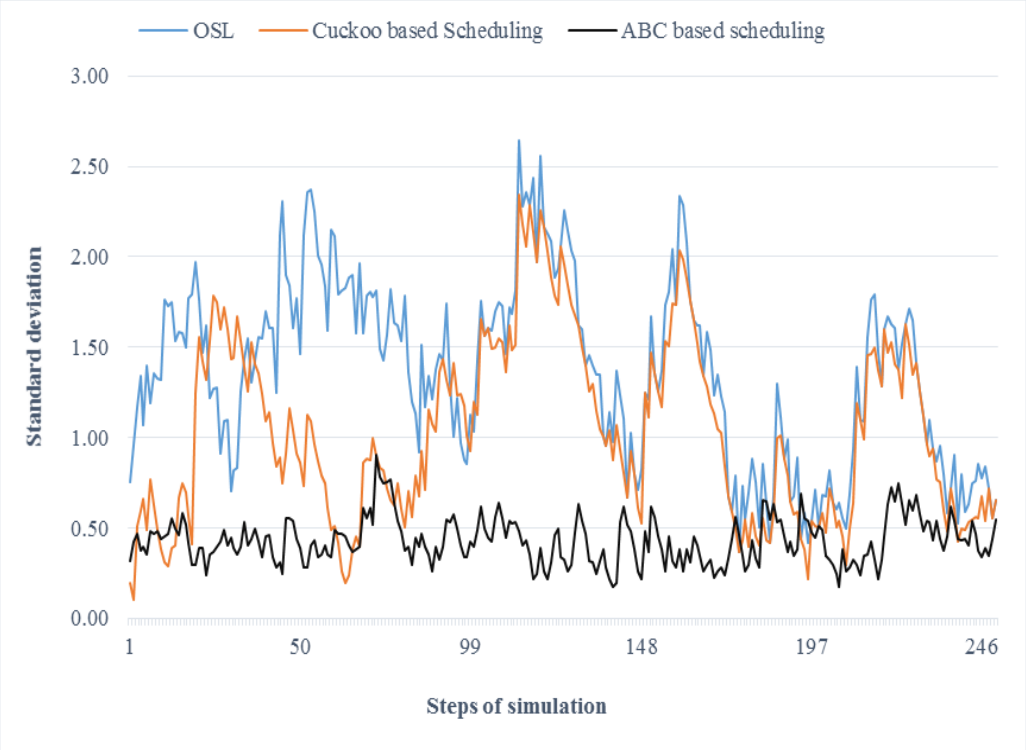

Figure 8. Standard Deviations of Makespan in each Step of the Simulation with 250 Resources and 100 Schedulers

As shown in the Figures above, in the second scenario, our proposed ABC based job scheduling method also outperform cuckoo based scheduling algorithm and OSL. With respect to the two scenarios mentioned above, it can be argued that since

the proposed method in this paper does not need to coordinate among the schedulers, hence, as the time passes, it will not change the amounts of loads. However, in OSL, unless all the schedulers are coordinated with each other, the load amounts of the resources will be highly variable. Moreover, it can be argued that the proposed method maintains load balance in both scenarios significantly better than OSL and cuckoo based scheduling algorithm.

5. Conclusion and Directions for Future Research

This paper focused on load balancing in cloud computing environment. The present paper introduced a novel method for decentralized and load-balanced job scheduling in grid environments. The proposed scheduling algorithm was developed on the basis of artificial bee colony. The purpose of proposing this new method was to maintain load balance in grid. Hence, it can be argued the proposed method used decentralized scheduling. However, in this method, the schedulers did not have any communications with each other and no information exchange was fulfilled among the schedulers. It is obvious that this will result in a reduction of network traffic. In this method, with respect to the magnitude of task load and the speed of each resource, a qualification value is measured and allocated to a certain resource. Therefore, tasks are allocated to the resources based on the qualification values. Then, after the completion of each task, the qualification value is updated and task allocation is fulfilled based on the new value. We conducted a series of simulation-based experiments in order to evaluate our proposed approach. The simulation was conducted in Matlab software and its results revealed that significant load balances were achieved in systems with decentralized scheduling. All the Figures were used to help depict the results and findings of the simulations. To sum it up, it can be argued that, based on schedulers' decisions, the proposed method appropriately divided and distributed tasks among resources and achieve desirable results. Consequently, the proposed method managed to achieve load balance in the respective grid environment.

References

[1] I. Foster, Y. Zhao, I. Raicu and S. Lu, "Cloud computing and grid computing 360-degree compared", in Grid Computing Environments Workshop. GCE'08, **(2008)**.
[2] M. Baker, R. Buyya and D. Laforenza, "Grids and Grid technologies for wide-area distributed computing", Software: Practice and Experience, vol. 32, no. 15, **(2002)**, pp. 1437-1466.
[3] A. Bouyer, M. Karimi, M. Jalali and M. N. M. SAP, "A new Approach for Selecting Best Resources Nodes by Using Fuzzy Decision Tree in Grid Resource Broker", International Journal of Grid and Distributed Computing, vol. 1, no. 1, **(2008)**, pp. 49-62.
[4] S.-H. Ko, N. Kim, J. Kim, A. Thota and S. Jha, "Efficient runtime environment for coupled multi-physics simulations: Dynamic resource allocation and load-balancing", in Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on, **(2010)**.
[5] J. Wu, X. Xu, P. Zhang and C. Liu, "A novel multi-agent reinforcement learning approach for job scheduling in Grid computing", Future Generation Computer Systems, vol. 27, no. 5, **(2011)**, pp. 430-439.
[6] B. Yahaya, R. Latip, M. Othman and A. Abdullah, "Dynamic load balancing policy with communication and computation elements in grid computing with multi-agent system integration", International Journal of New Computer Architectures and their Applications (IJNCAA), vol. 1, no. 3, **(2011)**, pp. 757-765.
[7] R. Garg and A. K. Singh, "Adaptive workflow scheduling in grid computing based on dynamic resource availability", Engineering Science and Technology, an International Journal, vol. 18, no. 2, **(2015)**, pp. 256-269.
[8] I. Al-Azzoni and D. G. Down, "Decentralized load balancing for heterogeneous grids", in Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD'09. Computation World:, **(2009)**.
[9] K.-Q. Yan, S.-C. Wang, C.-P. Chang and J. Lin, "A hybrid load balancing policy underlying grid computing environment", Computer Standards & Interfaces, vol. 29, no. 2, **(2007)**, pp. 161-173.

[10] Y. Li, Y. Yang, M. Ma and L. Zhou, "A hybrid load balancing strategy of sequential tasks for grid computing environments", Future Generation Computer Systems, vol. 25, no. 8, **(2009)**, pp. 819-828.
[11] J. Cao, D. P. Spooner, S. A. Jarvis and G. R. Nudd, "Grid load balancing using intelligent agents", Future generation computer systems, vol. 21, no. 1, **(2005)**, pp. 135-149.
[12] M. A. Salehi, H. Deldari and B. M. Dorri, "Balancing load in a computational grid applying adaptive, intelligent colonies of ants", Informatica, vol. 33, no. 2, **(2009)**.
[13] R. Buyya, D. Abramson and S. Venugopal, "The Grid Economy", Proceedings of the IEEE, vol. 93, no. 3, **(2005)**, pp. 698-714.
[14] M. Krystek, K. Kurowski, A. Oleksiak and K. Rzadca, "Comparison of centralized and decentralized scheduling algorithms using GSSIM simulation environment", in Grid Computing, **(2008)**.
[15] R. Ranjan, M. Rahman and R. Buyya, "A decentralized and cooperative workflow scheduling algorithm", in Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on, **(2008)**.
[16] S. P. Dandamudi, "Sensitivity evaluation of dynamic load sharing in distributed systems", IEEE concurrency, no. 3, **(1998)**, pp. 62-72.
[17] S. A. Elenin and M. Kitakami, "Performance analysis of static load balancing in grid", International Journal of Electrical & Computer Sciences IJECS/IJENS, vol. 11, no. 03, **(2011)**, pp. 57-63.
[18] A. Bouyer and B. Arasteh, "An Adaptable Job Submission System Based on Moderate Price-Adjusting Policy in Market-Based Grids", Wireless Personal Communications, vol. 73, no. 4, 2013/12/01, **(2013)**, pp. 1573-1590.
[19] M. Mezmaz, N. Melab and E.-G. Talbi, "An efficient load balancing strategy for grid-based branch and bound algorithm", Parallel computing, vol. 33, no. 4, **(2007)**, pp. 302-313.
[20] H. Mehta, P. Kanungo and M. Chandwani, "Decentralized content aware load balancing algorithm for distributed computing environments", in Proceedings of the International Conference & Workshop on Emerging Trends in Technology, **(2011)**.
[21] M. Rabiee and H. Sajedi, "Job scheduling in grid computing with cuckoo optimization algorithm", International Journal of Computer Applications, vol. 62, no. 16, **(2013)**, pp. 38-44.
[22] J. Taheri, Y. Choon Lee, A. Y. Zomaya and H. J. Siegel, "A Bee Colony based optimization approach for simultaneous job scheduling and data replication in grid environments", Computers & Operations Research, vol. 40, no. 6, 6//, **(2013)**, pp. 1564-1578.

Author

Asgarali Bouyer, he is an assistant professor in the faculty of computer engineering and information technology at Azarbaijan Shahid Madani University, Tabriz, Iran. He received Ph.D. degree in Computer Science from University of Technology Malaysia (UTM), in 2011. His research interests are in distributed computing (Grid and cloud computing), Data mining and complex networks. Application areas include social networks analysis, cluster computing and data mining, *etc*. He has been involved in several Malaysian and Iranian research projects.

