VOL. 12, NO. 20, OCTOBER 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

A NEW ADAPTIVE SCHEDULING METHOD FOR GRID COMPUTING

Ebrahim Aghaei
Department of Computer, Ramhormoz branch, Islamic Azad University, Ramhormoz, Iran
E-Mail: e.aghaei@iauramhormoz.ac.ir

ABSTRACT

By enlarging Grid, there is a requirement to use some new methods in order to manage it in various issues. One of the essential issues in the management of resources is scheduling. With regards to the dynamism and vast size of Grid, there is a need to adapt scheduling to surroundings and assign acts to resources so as to increase the efficiency and do jobs at an acceptable time. A method which is presented in this article is able to conforms itself with Grid after a quick discovering Grid changes and allocates jobs to resources to has an allowable efficiency in the current period, also it eventually decreases makespan. The proposal method is compared with other methods and the result of these experiments indicate the efficiency of this method toward other scheduling algorithms.

Keywords: grid scheduling, adaptable scheduling, adaptability, job scheduling.

1. INTRODUCTION

One of the mankind's wishes is accessibility to extremely fast and dense storage resources. We like having some resources which do our jobs within a minimum time or having ones which is able to store our unlimited files in itself in order to have an accessibility to those files if an unexpected problem occurs to our device.

In the past, researchers were going to produce a computer to calculate lots of users' computations. They wanted to produce one so that users could connect it easily at anytime and do their jobs. To produce such a computer, a lot of researches had been done but to producing this computer was impossible in reality. However, the researches of producing a computer which the users could connect it simply and do their computations didn't lead to producing a very powerful one but it led to discover a new technology that is named Grid. By this technology, it became possible that several resources could create such a calculating power with together. The resources that are connected via the network make us capable to create a coherent system which any resources are not able to create this storage and calculation potency. Nowadays, Grid provides ability for ordinary users so as to connect it simply and do their huge computations. It is the purpose of Grid which is inspired by the electricity network template, it means plug it in the outlet and use the limitless power of the electricity [1]

Grid confronts with a lot of challenges which are not possible to exist in other systems. Among the all possible challenges, some of them such as being large, various resources, too much changes, diverse policies and security are more important [2]. Grid can be either as much as a small local network or a global system that is connected by the internet of the resources [3]. There can be a lot of different resources in Grid. These resources can be either a minor simple resources like a sensor or a supercomputer in organization. Some resources are added to the system at each moment and some other resources are exited. The resources that get involved in defect and other resources will be repaired. The amount of a resource workload decreases and another one increases. A resource which is appropriate at the moment is inappropriate at another moment. The owners of the resources apply their diverse policies for cooperating in Grid. Some of them want to put upon via their resources and other ones want to cooperate in humanitarian activities. As for the resources are accessible for others, it is necessary to protect them from unforeseen dangers. On the other hand, it is possible that the users' programs are changed by that resourceso it is not desirable. These Grid's features have resulted in confronting the related issues with a great number of challenges.

There are many issues in Grid that researchers have been attempting to improve them by new methods. One of the most important issues is its effects on other existent issues in Grid. If the scheduling is done inaccurately, the system efficiency will decrease. So there is a need to find new methods to increase the efficiency of the system.

Nowadays, the scheduling is becoming more complicated by Grid growth [4]and it confronts with new challenges each day. The amount of the Grid changes has increased by its growth. Therefore, there is a requirement to find some methods to discover the changes more dynamic [5]. If the happened changes donot reach the scheduler in time, the scheduler cannot do the jobs accurately and it is possible that the scheduler assigns some jobs to resource which is not suitable at the moment and causes a reduction in the general efficiency of the system.

The structure of the article is formed to describe it after considering the previous jobs in part 2 of the Grid scheduling architecture. The structure of the suggested scheduling algorithm will be described in part 3 and the way of assigning the resources to jobs will be considered. The part 3 includes experiments and comparison of the suggested method with other existent methods. There is a conclusion in part 5.

2. Related works

In recent years, a lot of methods have been recommended for the jobs scheduling in Grid. Some articles have compared many scheduling methods. In [6]11 heuristics different methods and in [7] 13 different

heuristics scheduling methods is which the authors have analyzed and compared them with together after explaining the various heuristic methods.

A scheduling method on double layer line is suggested in [8]. In the first layer, jobs are assigned to a proportional machine according to a specified selective criterion and in the second layer; a different parallel scheduler is used to specify jobs to any resources.

Assigning the resources is done by learner agents in [9] and the authors described the multiagent strengthening learning. A multiagent scheduling method is recommended for the scheduling of Grid in [10].The agents are capable of adapting themselves to the incongruous and dynamic surroundings of Grid. The dynamic adaptability of the agents has been combined with static of the scheduler so as to leading to better results.

Some different evolutionary approaches are presented for the issue of the grid scheduling. In [11] authors have used an evolutionary and exploratory method, named General External Optimization (GEO), for the Grid scheduling, this method is formed by two steps. The first step is to assign jobs to appropriate resources and the second step is job scheduling in local resources independently. It is used for the grid scheduling in different resources of the genetic algorithm. The genetic algorithm which is based on Darwin theory uses selection, crossover and mutation operators in furtherance. Jiang and Chen [12] have used the genetic algorithm in their jobs. At first, the authors use a new initialization strategy to produce initial population and then use a set of new operators in order to achieve better performance. There is a new method which is based on PSO algorithm, has been used in [13]. In [14] a newfound method includes security issues to the super exploratory scheduling algorithm, has been used to schedule jobs that involves security issues. Bee Colony Optimization (BCO) algorithm is used in [15]. The authors have used a method according the smart binary bee colony algorithm. The balanced ant colony algorithm is used in [16].

Adaptability is one the necessary requirements in modern Grid environment. With regards to the great amount of incongruence in resources and jobs, also too much changes in Grid. Needing some new adaptation method is more felt [17]. App Les project is described [18] that it use a methodology for adaptive scheduling in applied programs. There are two developed algorithm in [19] which use predictive methods to schedule jobs on both system level and applied programs. In the scheduling of the applied programs, genetic algorithm is used to minimize the average time of accomplishing jobs, this action is done by allocating optimum to each node. In [20][21] some solutions are provided to add a self-management layer to software, it helps to adapt system to environment. An adaptive scheduling method for workflow according to the accessibility of dynamic resources is suggested in [22].

3. The Grid scheduling architecture

Grid scheduling architecture has user interface module, Global scheduler module and Grid Information System (GIS) module. All of these components share the needed information via Grid Information System (or database). The proposed scheduling system components are showed in Figure-1.

The user interface module provides an environment to make the users capable to deliver their applied programs to system and then receives necessary results. After receiving the applied programs of user, the user interface module turns it into one or more job and delivers it to the global scheduling system. The user interface module is a bridge between users and Grid.

Grid information system (GIS) module collects the information of the state of resources, received jobs, job queues and etc. In continue store them in the database of the system. The resource information can be the kind of architecture, operating system, speed, workload, number of processors, the amount of memory, communication bandwidth and etc. The existent information in this module must be alternatively updated. The updated information in this module have a vital role in the way of scheduling because if the information are not be updated and are delivered to scheduler wrongly, the scheduler will make wrong decisions. This issue will cause decrease the system performance. Therefore, the existent information in this part can be effective.

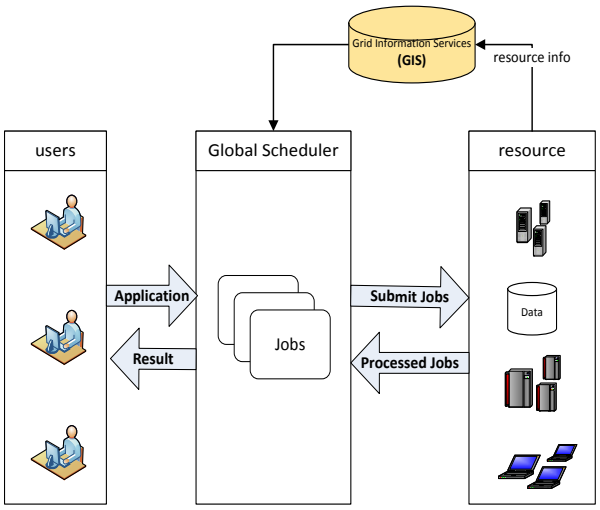

Figure-1. Grid scheduling architecture.

The scheduling module is divided into three sub module: job queue, scheduler and distributor. Each applied program is divided into one or more job. After delivering jobs, they enter the job queue and the scheduler appoints one or more job among the existent jobs in the queue and the distributor sends selected jobs to specified resources by the scheduler.

The job queue is used to store the ready jobs to run. This queue keeps the list of jobs which the system is not ready to run them at the moment (it is possible that

there are not enough available resources). Job queuing can be done according to different levels of job priority.

The sub module of the job scheduler is responsible for allocating the available resources which run the ready jobs. The scheduler selects the most appropriate resource to run job. Selecting a source is done according to the last state of the available resources which are achieved by Grid information system and entered jobs to system.

The distributor sub module sends job to a resource which it must run them. This sub module receives job from the scheduler and then sends them to the correspondent in order to run it in that resource (job migration).

4. The structure of the proposal system

The purpose of proposed adaptive scheduling is to allocate the users' applied programs to the most appropriate source in Grid. The structure of the system is showed in Figure-1.

4.1. The resources allocation

In Grid scheduling, it is assumed that m source is available R_1, R_2, R_3, …, R_m exist in Grid environment and n job, J_1, J_2, … , J_n received job must be scheduled in these resources.. The purpose is to schedule jobs in resources as the time of the completion jobs (makespan) must be minimized.

4.2. Proposed scheduling

In proposal scheduling, the speed of each resource and workload in different time periods are received by GIS so that the $_{scheduler}$ makes better decisions according to them. The purpose is to adapt the scheduler to changes and current conditions of Grid environment. We show the speed of resource R_i by S_i and the workload of it by W_i. The scheduler receives the workload and current speed of resources by GIS in any time period and arranges them non-descending according to gathered information and current capacity of resources. According to the gathered information, the capacity of each resources can be calculated:

Equ.1 $C_i = S_i * (1- W_i)$

In equation 1, S_i is the speed of resource R_i in time period a to b, W_i is the workload of resource R_i in time period a to band C_i is the current capacity of resource R_i. The state of resources can be discriminated by C_i and the scheduler can adapt itself to environment by it.We arrange resources non-ascending according to their current capacity and then arrange jobs for non-ascending. We allocate the first job to the first resource in list and similarly allocate a job to any resources. After having done this process, we calculate makespan of all resources.

After calculating the makespan of all resources, the max makespan will be obtained. When the max makespan is calculated, some jobs will be allocated to resources with less makespan so that their makespan approximately become as much as max makespan. Assigning jobs in this step is from max to min. Assigning jobs to resources will repeat until the time period does not reach at the end.

Equ.2 $maxI_i = S_i * (maxMakespan - Makespan_i)$

In Equation 2, max I_i is the maximum number of commends which can be run in current time period in the resource R_i sothat the makespan of this resource becomes the worst makespan of current resource. MaxMakespan is the maximum makespan among all resources in current time period and $Makespan_i$ is the makespan of the resource R_i in current time period.

After calculating the maximum instructions of each resource in current time period, jobs are non-ascending allocated to resource. If the number of a job instruction is more than the maximum instructions of a resource, we will go to next job in list. Allocation of jobs is from the beginning to the end of the list.

The proposal algorithm such acts that if a resource workload exceeds a specified range (for example 95%), no job will be allocated to it and this process will repeat until the resource workload returns to an acceptable level.

In continue we present a sample issue so in order that we explain the proposal method much more by it. In this example, we assume that there are 14 jobs at the start of the algorithm. You can see jobs length in Figure-3. In this example, we suppose that four jobs are assigned to resources in any time period. Also, at first we suppose that four resources R_1, R_2, R_3, R_4 are available ($R = \{R_1, R_2, R_3, R_4\}$). The speed of these resources at the start of the job are respectively 20, 10, 15, 5 ($S= \{20, 10, 15, 5\}$).

computes Capability of each resource;
sortedC= sort of resource basis Capability in descend order;
SortedJ=sort of job in descend order;
while (there are Jobs to schedule)
{
if(period is finished)
{
receives resource information from GIS;
computes Capability of each resource;
sortedC= sort of resource basis Capability in descend order;
}
while (for all resource in sortedC)
{
allocate resource i to Job j;
i=next resource in sortedC;
remove current j from sortedJ;
j=next job in sortedJ;
}
compute makespan for any resource;
maxMakespan=maximum($makespan_i$)
while (for all resource in sortedC)
{
$maxI_i = S_i * (maxMakespan - Makespan_i)$
while ($jobSize_j < maxI_i$)
{
allocate resource i to job j;
remove current j from sortedJ;
j=next job in sortedJ;
}
}

Figure-2. Proposal algorithm.

The scheduler received the necessary information (for instance: speed and workload of resources) by GIS. You can see the obtained information in time period t_0 to t_1 in Figure-3.

After having received the workload (W) and speed (S) of resources, the scheduler calculates the current capacity of resources (C). Then it arranges resources non-ascending. If the first resource in list is able to run a job, the job will be allocated to that resource but if it is not able to run a job, next resource will be experimented. This resource will be repeated until a job will be allocated to a resource in list. It should be noted that if we reach the last resource in the list of resources I a time period, we will obtain the makespan of all resources. Then the maximum is considered to allocate job to other resources so that their makespan approximately become as much as makespan. After this process, algorithm repeats from beginning to schedule all jobs.

Resources allocation in time period t_0 to t_1 is shown in Figure-3. As you see, the biggest job in list (J12) is allocated to the first resource in the list of arranged resources which based on the current capacity (R_1). The second biggest job (J_3) will be allocated to next resource in the list of resources (R_3). Job J_8 to the third resource and job (J_{14}) to the fourth resource will be allocated.

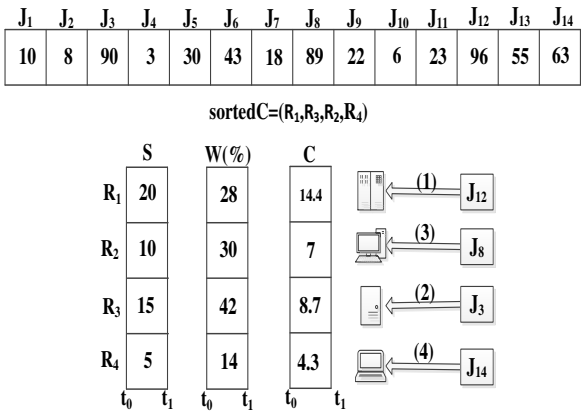

Figure-3. Resources allocation at the time t_0 to t_1 by the proposal method.

Then makespan of all resources will be calculated to obtain the makespan. After calculating it, some jobs will be allocated to resources with less makespan so that their makespan approximately become as much as the max makespan. Job allocation in this step is from max to min. Allocating jobs to resources repeats until the time period is not finished. In the example of the Figure 3, the resources makespan respectively becomes 6.67, 12.71, 10.34 and 14.65 by allocating jobs to these resources. Therefore, some jobs will be allocated to the first to third resources in order that their makespan almost becomes the size of 14.65. To do this, we use equation 2.

$maxI_1 = 14.4 * (14.65 - 6.67) = 114.91$
$maxI_2 = 7 * (14.65 - 12.71) = 13.58$
$maxI_3 = 8.7 * (14.65 - 10.34) = 37.49$

These numbers suggest that the first resource with a maximum of 114.91 instructions, the second one with a maximum of 13.58 instructions and the third one with a maximum of 37.49 instructions can run in order that their makespan becomes like the fourth resource which has the max makespan in this period. Then we allocate jobs from max to min to the first resource of the list. If allocating a job to resource causes increases the total value from the obtained value (for the first resource 114.91), that job will not be allocated to resource, and algorithm will go to next job. This process will be repeated till we achieve the desired total value or there are not any jobs to assign to this resource. This operation is performed for the following resources. The allocation of Jobs becomes the form of Figure-4 after accomplishing the operation.

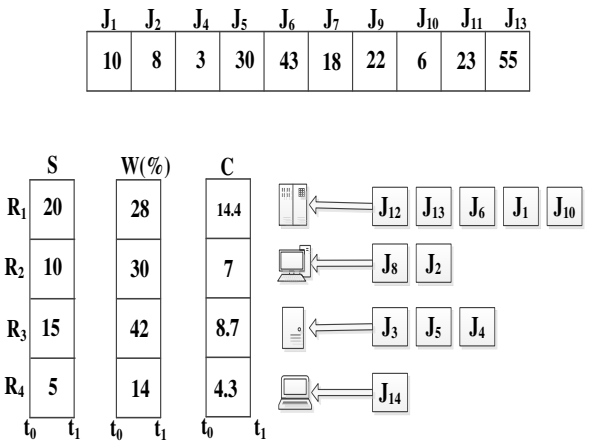

Figure-4. Continuing the resources allocation at the time t_0 to t_1 by the proposal method.

As expected and in Figure-4 you can see that more jobs are allocated to the first resource which has more capacity.

In the next time period, other jobs will be allocated according a resource which is described in the current period. If a resource is detected as a disabled one or its workload has increased, no jobs will be allocated to that resource because of allocating job to resources like this one deceases the system efficiency.

5. Experiments

We consider the efficiency of the proposal method in this part. The proposal method is attempting to adapt itself with the best possible way as for the existent current conditions in Grid. Our purpose is to present a method for scheduling to schedule the entered independent jobs in such a way that either jobs are run faster or the efficiency of resources is increased.

To simulate and valuate of the proposal method, we have simulated grid's environment with several independent jobs and incongruous resources. Job length in terms of million instructions (MI) and resources' speed in terms of million instructions per second (MIPS) are expressed.

The users' specified programs are divided into one or several more independent job and are entered to job queue. We consider the number of jobs from 1 to 10000 in different experiments. Jobs length in a specified period is randomly made in each experiment. We consider the job length range of 1000 to 100000000. The speed of each resource in range of 1000000 to 100000000 is randomly specified. Also time periods in range of 0.001 to 0.01 second is changing.

Workload is also used in the proposal method, because of this reason the workload of each resource is accidently specified in range of 1 to 100 in each time period. The produced workload for each resource is suggested in terms of percentage. If a resource's workload is obtained 20, it means only 20% of the source's capacity is being using and the remaining 80% is not being used at the moment. If a resource's workload is set 100, it means 100% of its capacity is being used.

We compare our proposal method with some good and known algorithm and the results are presented in Figure-5 to Figure-8. Our estimation according to makespan is the execution time and the average utilization. Because some data are random, the experiments have been done for 50 times for each set of data and their average is showed.

In the experiments, we have increased the number of jobs from 100 to 1000 and considered the job length in the range of 1000 to 100000 and the speed of the resource in the range of 1000000 to 1000000000.It can be observed by the results, whatever the incongruence, number of jobs and resources are increased, the proposal method obtains better results.

Another noticeable point about these figures is that the proposal method gets involved in less fluctuation because of considering the workload in scheduling action while other methods do not consider it. It mean it is possible that other method in some operations assign more jobs to resources with high workload and assign more jobs to resources with low workload in some other operations.

As you see in Figure-5 the least makespan is related to the proposal method.

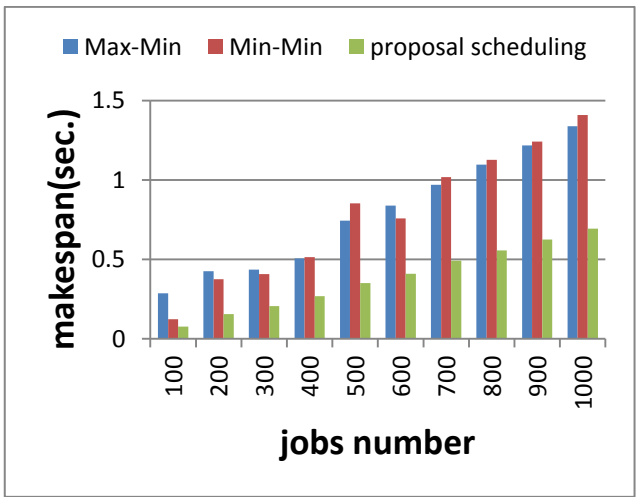

Figure-5. Algorithm comparison according to makespan. We have considered the number of resources 5 in Figure-6 and the numbers of jobs are changing from 1000 to 10000.

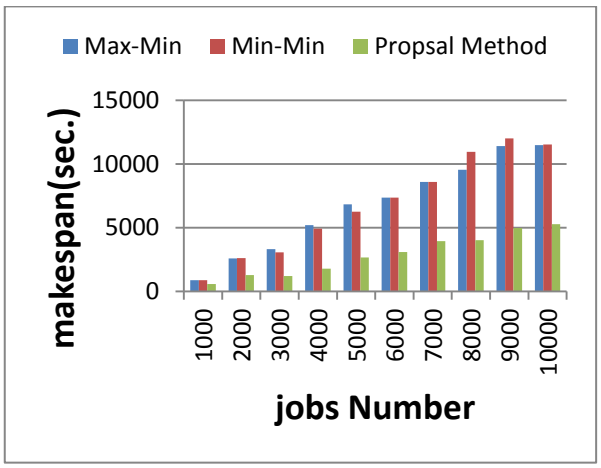

Figure-6. Algorithm comparison according to makespan.

In another experiment that you can see its results in Figure-7, we consider the number of jobs constantly (we considered them 1000) and increased the number of resources in each step (the number of them is has been increased from 2 to 10). As you see in the Figure-7, the proposal method obtains less makespan.

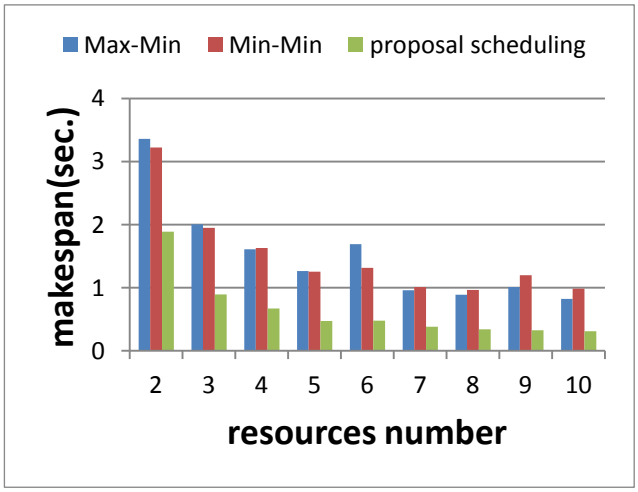

Figure-7. Algorithm comparison according to makespan.

Another important criterion in Grid algorithm is resources' efficiency. The purpose of this criterion is that all available resources are being optimally used. It is possible to use this criterion in order to show workload balanced in system. We calculate the arrange utilization of the resources in question 3:

*Equ3.*averageUtlization= $\frac{\sum completion[r]}{makespan*resourceNumber}$

In Equation 3, completion[r] is the completion of the last performed job in each resource, makespan is thebiggest completion time among resources and the resourceNumber is the total number of resources.

As you see in Figure-8, proposal method presents a better average utilization from resources toward other methods. In other words, it balances the workload in system much better. The reason is that our proposal method utilizes the most appropriate resources in each time period and does not utilize resources with less efficiency.

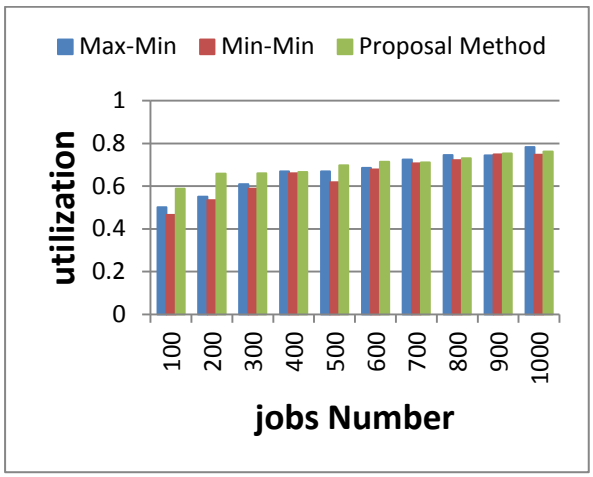

Figure-8. Algorithm comparison according to the resources' average utilization.

6. CONCLUSIONS

One of the most important issues in grid scheduling is that the scheduler increases the system efficiency by allocating suitable resource to jobs appropriately. To allocate resources to jobs suitably, needing the exact information of resources and jobs at deciding time is necessary. The changes must be discovered in time and an acceptable scheduling is done. With regards to the size and so much changeability in Grid, a method is used in this article to discover changes in different alternative time period and the scheduler uses these changes. The resources will be estimated again after the initial scheduling and the scheduling will be done again according to their capacity so as to utilize all capacities of resources. This method is compared with other scheduling methods and shows the results of efficiency improvement.

ACKNOWLEDGMENT

Funding support of this research was provided by the Ramhormoz branch, Islamic Azad University, Ramhormoz, Iran. Moreover this paper was derived from a report named “a new adaptive scheduling method for Grid computing”.

REFERENCES

[1] I. Foster and K. Kesselman. 2004. The Grid 2: Blueprint for a New Computing Infrastructure, 2nd ed., Morgan Kaufmann Publishers.

[2] F. Xhafa and A. Abraham. 2010. Computational models and heuristic methods for Grid scheduling problems. Future Generation Computer Systems. 26(4): 608-621.

[3] F. Dong and S. G. Akl. 2006. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. Technical Report No. 2006-504.

[4] M. B. Qureshi, M. Mehri Dehnavi, N. Min-Allah, M. S. Qureshi, H. Hussain, I. Rentifis, N. Tziritas, T. Loukopoulos, S. U. Khan, C.-Z. Xu and A. Y. Zomaya. 2014. Survey on Grid Resource Allocation Mechanisms. Journal of Grid Computing.

[5] X. Zhang, C. Germain and M. Sebag. 2010. Adaptively detecting changes in Autonomic Grid Computing. 11th IEEE/ACM International Conference on Grid Computing. pp. 387-392.

[6] T. D. Braunt, H. J. Siegel and *et al*. 2001. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing. 61(6): 810-837.

[7] A. A. Chandio, K. Bilal, N. Tziritas, Z. Yu, Q. Jiang, S. U. Khan and C.-Z. Xu. 2014. A comparative study on resource allocation and energy efficient job scheduling strategies in large-scale parallel computing systems. Cluster Computing. pp. 1349-1367.

[8] A. Tchernykh, U. Schwiegelshohn, R. Yahyapour and N. Kuzjurin. 2010. On-line hierarchical job scheduling on grids with admissible allocation. Journal of Scheduling. pp. 545-552.

[9] A. Galstyan, K. Czajkowski and K. Lerman. 2005. Resource Allocation in the Grid with Learning Agents. Journal of Grid Computing. pp. 91-100.

[10] T. Altameem and M. Amoon. 2010. An Agent-Based Approach for Dynamic Adjustment of Scheduled Jobs in Computational Grids. Journal of Computer and Systems Sciences International. 49(5): 765-772.

[11] P. Switalski and F. Seredynski. 2015. Scheduling parallel batch jobs in grids with evolutionary metaheuristics. Journal of Scheduling. pp. 345-357.

[12] Y.-S. Jiang and W.-M. Chen. 2015. Task scheduling for grid computing systems using a genetic algorithm. The Journal of Supercomputing. pp. 1357-1377.

[13] H. Izakian, B. Tork Ladani and *et al*. 2010. A Discrete Particle Swarm Optimization Approach for Grid Job Scheduling. International Journal of Innovative Computing, Information and Control. 6(9): 4219-4233.

[14] R. Aron, I. Chana and A. Abraham. 2015. A hyper-heuristic approach for resource provisioning-based scheduling in grid environment. The Journal of Supercomputing. pp. 1427-14-50.

[15] S.-S. Kim, J.-H. Byeon, H. Liu, A. Abraham and S. McLoone. 2012. Optimal job scheduling in grid computing using efficient binary artificial bee colony optimization. Soft Computing.

[16] R.-S. Chang, J.-S. Chang and P.-S. Lin. 2009. An ant algorithm for balanced job scheduling in grids. Future Generation Computer Systems. pp. 20-27.

[17] M. Botón-Fernández, M. Rodríguez-Pascual, M. A. Vega-Rodríguez, F. Prieto-Castrillo and R. Mayo-García. 2014. A Comparative Analysis of Adaptive Solutions for Grid Environments. International Journal of Parallel Programming.

[18] F. Berman, R. Wolski and *et al*. 2003. Adaptive Computing on the Grid Using App LeS. IEEE Transactions on Parallel and Distributed Systems. 14(4): 369-382.

[19] Y. Gao, H. Rongb and J. Z. Huangc. 2005. Adaptive grid job scheduling with genetic algorithms. Future Generation Computer Systems. 21(1): 151-161.

[20] M. Salehie and L. Tahvildari. 2009. Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems. 49(2): 1-42.

[21] R. Nou, F. Julia and *et al*. 2011. A path to achieving a self-managed Grid middleware. Future Generation Computer Systems. 27(1): 10-19.

[22] R. Garg and A. K. Singh. 2015. Adaptive workflow scheduling in grid computing based on dynamic resource availability. Engineering Science and Technology an International Journal.