Computer Engineering and Applications Vol. 5, No. 2, June 2016

A Modified Black hole-Based Task Scheduling Technique for Cloud Computing Environment

Fatemeh ebadifard [1], Zeinab Borhanifard [2]

[1]*Department of computer, Iran University of science and technology, Tehran, Iran*
f_ebadifard@comp.iust.ac.ir
[2] *Department of computer, Qom University, Qom, Iran*
borhanifard.zeinab@gmail.com

ABSTRACT

The issue of scheduling is one of the most important ones to be considered by providers of the cloud computing in the data center. Using a suitable solution lets the providers of cloud computing use the available resources more. Additionally, the satisfaction of clients is met through provision of service quality parameters. Most of the solutions for this problem aim at one of the service quality factors and in order to achieve this goal, variety of methods are used. Using the algorithm of modified black hole in this paper, a proper solution is presented to tackle the problem of scheduling the affairs in cloud environment. The proposed method reduces makespan, increases degree of load balancing, and improves the resource`s utilization by considering the capability of each virtual machine. We have compared the proposed algorithm with existing task scheduling algorithms. Simulation results indicate that the proposed algorithm makes a good improvement regarding the makespan and amount of resource utilization compared to schedulers based on Random assignment and particle swarm optimization Algorithms.

Keywords*:* cloud computing, task scheduling, Black hole, makespan, resource utilization.

1. INTRODUCTION

Cloud computing is one of the fields that has drawn attention of lots of the users in recent years. This is due to significant advantages that cloud services prepare for users in terms of cost and efficacy. The cloud environment provides a bed of servers in data center to provide the users sharing them as soon as they request for the resources. Service providers can provide the users with variety of services, renting virtual machines of cloud providers. Since providers of different services achieve the necessary virtual machines through cloud providers, the basic challenge of the service providers is presenting an effective method for scheduling the tasks in a way that they can provide the service quality necessary factors of aligned with needs of the service providers and users. Cloud task scheduling means optimized allocation of the requests to the computational resources available in data centers. When taking about scheduling, different kinds of virtual machines with specified constrains are presented for the users and service providers. Generally speaking, the scheduling algorithms are divided to two categories of static and dynamic ones:
Static scheduling: in algorithm of static scheduling, allocating tasks to the virtual machines takes place based on the capabilities of virtual machines and the primary status of each machine. In another words, this process is only based on primary

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

information related to the nodes and their characteristics. This information include the amount of processing power, internal storage and the capabilities of storing and other the power of integration among other virtual machines as well. The important feature of the dynamic algorithms is that these algorithms do not regard all of the changes taking place dynamically in virtual machines. Moreover, they do not have are not adapted to the change of work load in virtual machines over time.

Dynamic scheduling: unlike static algorithms, in dynamic methods, in addition to primary capabilities of the each virtual machine, they assign the tasks to the virtual machines based on the existing status of that machine and work load assigned to it and according to the results of these evaluations, they transfer the requests from a machine to another one. Although these methods are more complicated than static method but they have more efficacies [1].

Using the black hole algorithm in this paper [2] improved by the Simulated Annealing, a method is presented for scheduling the requests on virtual machines that in addition to the reducing the makespan time, increases the resource utilization. The advantage of the presented algorithm in above method comparing the previous ones is its simplicity and efficacy. The simulation results indicate that comparing the other heuristic algorithm such as PSO, above method could have better improvement in the makespan besides the efficacy of the resources. Briefly one can say that our main concentration in this paper includes the following:

- Presenting a suitable method for scheduling the requests using the modified black hole algorithm via simulated annealing, aiming at reduction of the makespan as well as Increase the utilization of the resources
- Using the fitness function to distribute the requests fairly
- The purposeful analysis and evaluation to show the amount of the efficacy of black hole algorithm in comparison to the other heuristic algorithms such as PSO

The rest of the paper consists of the following sections: in section 2 the related works are investigated; in section 3, after introducing the black hole algorithm, the mathematical model of algorithm is defined and then details of the suggested method are shown in details. In section 4 the simulation and evaluation of the suggested method are defined and finally in section 5, the conclusion and future works are presented.

2. RELATED WORK

In cloud computing terminology, optimal assignment of requests to data center recourses is called task scheduling. Requests are assigned to different kinds of resources considering the service they might need. Task scheduling is one of the most important problems in the cloud computing. There are many studies regarding scheduling of tasks in cloud computing. In what follows, we will discuss some of these methods.

In 2010, Fang [5] proposed a scheduling method with the goal of increasing load balancing. In the proposed method, scheduling is performed on two level. On the first level, tasks are sent to appropriate virtual machines and if the machine is not efficient enough, it is placed on an appropriate physical machine. Simulation results indicate that this method provides good improvements in makespan and utilization of processor.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

In 2010, Wang [3] proposed a method in which task scheduling is performed by a combination of Opportunistic Load Balancing (OLB) and Min-Min Load Balancing (MMLB). The combination of these two method reduces execution time and improves load balancing in the system. Moreover, the min-min scheduling algorithms minimizes the execution time of a task, which is possible through reducing the execution time of all tasks. The combination of these two algorithms maximizes efficient resource usage and increase the performance of task execution.

In 2012, Krishna et al. [6] proposed a scheduling method based on load balancing with the goal of reducing waiting time and increasing response time. The proposed method was inspired by the honey bee behavior. After assigning tasks, the proposed method divides virtual machines into three groups of under loaded, balance, and over loaded and if a machine is over loaded, tasks are transferred from that machine to under loaded one. Simulation results indicate that this method makes good improvements in makespan and response time. Moreover, it increases the degree of load balancing.

In 2012,Zhan et al. [12] proposed a mixed method using both PSO and Simulated Annealing (SA) algorithms. The proposed method tends to reduce tasks' average execution time and increase convergence speed to the optimal solution. This study shows that the improved PSO-based method has better efficiency comparing to genetic algorithm, simulated annealing, and ant colony; although, combining PSO and SA algorithms results in more computational complexity. Mixing PSO with SA algorithms, Kaur and Sharma [8] proposed a new task scheduling method for cloud environment. Their primary goal was to optimize resource utilization and maximize providers' profit.

In 2014,Abdi et al. [11] proposed an improved PSO-based task scheduling method using "shortest job to fastest processor" algorithm for generating the initial population. The method's goal was to reduce makespan time. Their results show that the proposed method has lower makespan time comparing to GA-based and simple PSO-based solutions.

In this paper, we propose a Black hole-based task scheduling algorithm by benefiting from SA algorithm to generate a more suitable initial population and choosing a more appropriate goal function. Our study shows that Black hole method has not been used to solve the problem of task scheduling in the cloud computing. Despite all the above mentioned works which consider either clients' benefits or providers' benefits, our method decreases makespan duration and increases resource utilization at the same time, and it can meet providers' and clients' needs simultaneously.

3. MODIFIED BLACK HOLE BASED TASK SCHEDULING

The Black hole-based task scheduling method is explained in this section. To bring a Mutual benefit to both providers and clients, the proposed method aims to maximize resource utilization and minimize the makespan time. First, we have briefly described Black hole technique and its components; then, we have presented the problem formulation and our proposed method in details.

3.1 CLASSIC BLACK HOLE

Black hole is a new meta-heuristic algorithm introduced by Hatamlou [2] at 2012. It is a population-based method that has some common features with other population-based methods. The BH algorithm is more similar to the natural black hole phenomenon and evolving of the population is done by moving all the candidates towards the best candidate in each iteration, namely, the black hole.

Like other population-based algorithms, in the proposed black hole algorithm (BH) a randomly generated population of stars can be considered as a possible solution for the problem, and it searches the problem space for the optimized solution. Using a fitness function, performance of each star is evaluated at the end of every iteration according to Eq. (3-6) and the best candidate in the population, which has the best fitness value, is selected to be the black hole and the rest form the normal stars. After initializing the black hole and stars, all the stars start moving towards the black hole according to Eq. (3-1).

(3-1)

$$x_i(t+1) = x_i(t) + rand \times (x_{BH}-, x_i(t)) \quad i = 1, 2, \ldots, N$$

Where xi(t) and xi(t + 1) are the locations of the ith star at iterations t and t + 1, respectively. xBH is the location of the black Hole in the search space. Rand is a random number in the interval [0, 1]. N is the number of stars.

In Every iteration if a star reach a location with lower fitness than the black hole, this star replace with black hole and then stars start moving towards this new location. Every star (candidate solution) that crosses the event horizon of the black hole will be sucked by the black hole. Every time a candidate (star) dies another candidate solution (star) is born and distributed randomly in the search space. The radius of the event horizon in the black hole algorithm is calculated using the following equation:

(3-2)

$$R = \frac{f_{BH}}{\sum_{i=1}^{N} f_i}$$

Where fBH is the fitness value of the black hole and fi is the fitness value of the ith star. N is the number of stars (candidate solutions).

When the distance between a candidate solution and the black hole (best candidate) is less than R, that candidate is collapsed and a new candidate is created and distributed randomly in the search space. The pseudo code of Modified black hole task scheduling is presented in Figure 1.

```
Initialize a population of stars with random locations in the search space

Loop

For each star, evaluate the objective function

Select the best star that has the best fitness value as the black hole

Change the location of each star according to Eq. (3-1)

If a star reaches a location with lower cost than the black hole, exchange their locations

If a star crosses the event horizon of the black hole, replace it with a new star in a random location
```

FIGURE 1 The black hole algorithm pseudo code [2]

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

3.2 TASK-RESOURCE SCHEDULING FORMULATION

Different objectives can be considered for scheduling tasks in cloud computing environment. Minimizing the makespan time and maximizing resource utilization is our focus of attention in our method.

To formulate the problem we have denoted a set of tasks $\text{Task} = \{T_1, T_2, T_3, \dots, T_i\}$ where $i \in \{1, 2, \dots, n\}$. Tasks are assumed to be non-preemptive and independent. We have defined a set of m virtual machines, $VM = \{VM_1, VM_2, VM_3, \dots, VM_j\}$ interconnected by network where $j \in \{1, 2, \dots, m\}$. The tasks will be processed on virtual machines. Completion time and processing time of task T_i on virtual machine VM_j are denoted as CT_{ij} and PT_{ij} respectively. The objectives are minimizing the overall task completion time and maximizing the average resource utilization. overall task completion time is called makespan and is defined by Eq. (3-3) which is extracted from [6]:

$$\text{Makespan} = \max\{CT_{ij} \mid i \in T, i = 1, 2, \dots, n \text{ and } j \in VM, j = 1, 2, \dots, m\} \qquad (3-3)$$

Virtual machines have its own processing unit, and processing time of each specific task on each specific VM is supposed to be known; therefore, utilization of each resource is defined by Eq. (3-4), and average utilization is defined by Eq. (3-5):

$$\text{Utilization}_{VM_j} = \frac{\sum_{i=1}^{n} PT_{ij}}{\text{makespan}} \qquad (3-4)$$

$$\text{Average Utilization} = \frac{\left(\sum_{j=1}^{m} \text{Utilization}_{VM_j}\right)}{m} \qquad (3-5)$$

Regarding our objectives, the fitness function is defined by Eq. (3-6):

$$\text{Fitness Function} = \frac{(\min)\ \text{makespan}}{(\max)\ \text{Average utilization}} \qquad (3-6)$$

Eq. (3-6) shows that a star has a better position if it has a lower fitness value, and this star can increase average resource utilization and decrease makespan time. The details of the proposed method is explained in the next subsection.

3.3 Proposed Black hole-Based Task Scheduling Method

Detailed explanation of the proposed method and pseudo code of its algorithm are presented in this section. The steps of the algorithms are as following:

Step 1: Defining of Stars and position vectors for a problem with n tasks, we have defined each star as a N-dimensional vector $X = (X_1, X_2, \dots, X_n)$, where $X_i \left(i \in \{1, 2, \dots, n\}\right)$ represents index of a virtual machine on which task i will be processed. Position of stars are initialized randomly. For example for a 6 task and 3 virtual machine problem a star's position vector can be initialized as shown in Table. 1.

TABLE 1

Values of a randomly initialized star

Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
VM 2	VM 1	VM 1	VM 3	VM 2	VM 2

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

Step 2: In standard Black hole algorithm initial stars are created randomly, but randomness decreases the chance of algorithm to converge to best solution, in order to improve the behavior of Black hole algorithm, we merge Simulated Annealing algorithm(SA) into Black hole, i.e. instead of generating initial population randomly we Improve them considering SA algorithm. All other steps are similar to standard Black hole algorithm.

Step 3: Calculating fitness function and Specifying black hole
Fitness value of each Star will be calculated using Eq. (3-6). Comparing the current fitness value of the whole population together, the lowest fitness value will be specified as black hole best position.

Step 4: Updating stars' position
The position vector of each star will be updated respectively. All the stars start moving towards the black hole according to Eq. (3-1).

Step 5: Terminating condition
Steps 3 and 4 will be repeated until the maximum number of iterations is reached.

Pseudo code of our proposed algorithm is shown in Fig. 2. Position initialization of our method has an advantage over the base Black hole task scheduling method because base Black hole task scheduling method initializes stars' position vectors randomly, but our proposed method executes a load balancing step after random initialization to improve the stars' positions. Therefore, each initial star can present a proper solution which will be improved by moving in the problem space. The simulation results, presented in section 4, show that comparing to the base black hole and the PSO-based task scheduling method, our method leads to more efficient completion time and resource utilization.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

Input: $VM = \{VM_1, VM_2, VM_3, ..., VM_j\}$, $Task = \{T_1, T_2, T_3, ..., T_i\}$

Output: best position of Tasks on the VM

Start:

1: Set star dimension equal to the size of ready tasks, Initialize stars position randomly.

2 : for each star run Simulated Annealing algorithm for balancing star position using fig 3

3: For all Stars, calculate its fitness value by in Eq. 3-6

If (fitness value < bh-fitness)

set the current fitness value as the new bh-fitness

4: For all stars, update their positions using Eq. (3-1)

5: For all Stars, calculate R= distance between star positon and blackhole position and R by using Eq. (3-2)

If (result<R)

Remove current star and replace it with a new star in a random location in the search space

6: repeat steps 3 to 6 If termination criteria or maximum iteration is not satisfied.

Figure 2 The Modified black hole-based task scheduling psuedo code

```
Procedure Simulated Annealing
 S = Choose an initial solution
 T = Choose an initial temperature
 REPEAT
  S' = Generate a neighbor of the solution S
  ΔE = objective(S') – objective(S)
  IF (ΔE > 0) THEN // S' better than S
   S= S'
  ELSE with probability EXP (ΔE/ T)
   S = S'
  END IF
  T = lower the T using linear/ non-linear techniques
 UNTIL meet the stop criteria
End
```

FIGURE 3 The Simulated Annealing psuedo code

4. SIMULATION RESULT

In this section, we have presented the simulation setup and evaluation of the proposed algorithm based on the results of conducted tests. It is worth to mention that the experiments are conducted in SaaS level. We have used CloudSim [14] to simulate cloud computing SaaS level. This simulator gives capability of simulating a virtualized environment, and it supports on demand resource provisioning. We have extended the CloudSim simulator for modeling our method. The purpose of our method, as mentioned before, is to present an efficient Black hole-based task scheduling algorithm for cloud computing environment which can reduce completion time of the longest task (makespan) and increase the average resource utilization of the cloud data center. We have defined 10 stars for the Black hole population. The termination condition is set to 100 iterations. To analyze our method, several experiments are conducted in two different test setups. Test setups and experiments are described in the rest of this section. The input parameters and variables used in the task scheduling problem are presented in Table 2.

TABLE 2.
The input parameters and variables for problem

Particle Swarm Optimization algorithm(PSO)	initial population	10
	Iteration Number	100
	C_1 و C_1	1.49445
	r_1 و r_2	Random number between 0 and 1
Black hole algorithm	initial population	5
	Iteration Number	100
	rand	Random number between 0 and 1
Simulated Annealing algorithm(SA)	Iteration Number	100
	Initial Temp	1000

- **Test setup 1**

In the first test setup, we have defined a cloud data center which has three hosts, each capable of supporting virtualization technology and sharing its resources among several virtual machines. Hosts' hardware specifications are presented in table. 3. Sixteen virtual machines are supposed to be running on these three hosts; each of them has a distinct specifications. Each virtual machine executes applications with different number of instructions varying from 500 to 4500. We have used simulated workload for these series of tests.

TABLE 3

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

Hosts' technical specifications

HostId	Number of processing Cores	Processing speed (Mips)	Ram (MB)	Hard (MB)	Bandwidth (Mbps)
1	4	5000	204800	1048576	102400
2	2	25000	102400	1048576	102400
3	1	10000	51200	1048576	102400

The results of the proposed method are compared to four other methods: 1) simple Round Robin (RR) algorithm, 2) classic PSO-based task scheduling method which initializes particles' position and velocity vectors randomly, 3) classic Black hole method, and 4) Modified Black hole -based method which aims to improve Black hole algorithm convergence speed taking advantageous from SA algorithm.

Fig 4. Compares makespan of these four methods for different number of tasks. The result illustrates that our proposed method outperforms other four methods. Our method is qualified to balance the load of Stars in the beginning of first iteration, therefore it provides a better makespan comparing to the other methods specially with more tasks. The proposed method affects the makespan time more effectively than others.

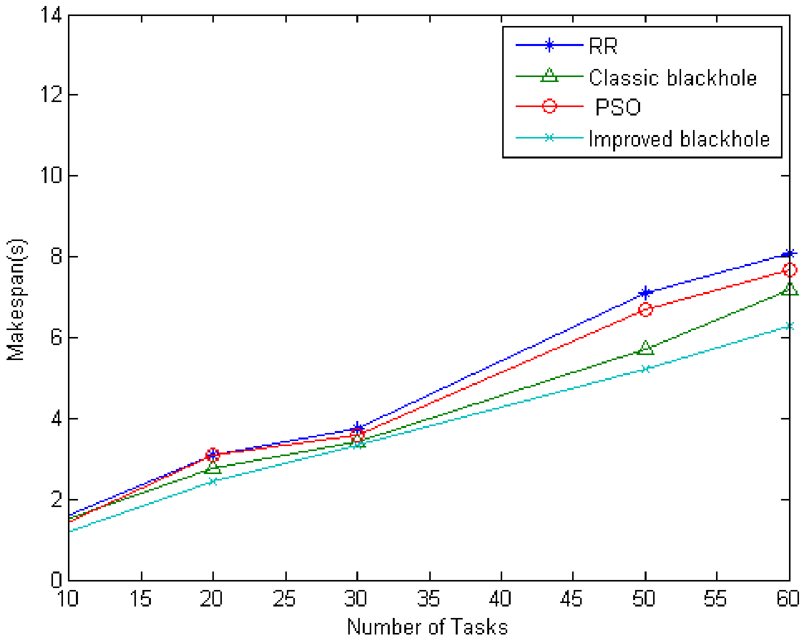

FIGURE 4. Makespan Comparison

Fig. 5 represents response time for different number of task for the Round Robin, PSO, classic Black hole, and the proposed algorithms. It is clear that our method provides a better response time in comparison to three other methods.

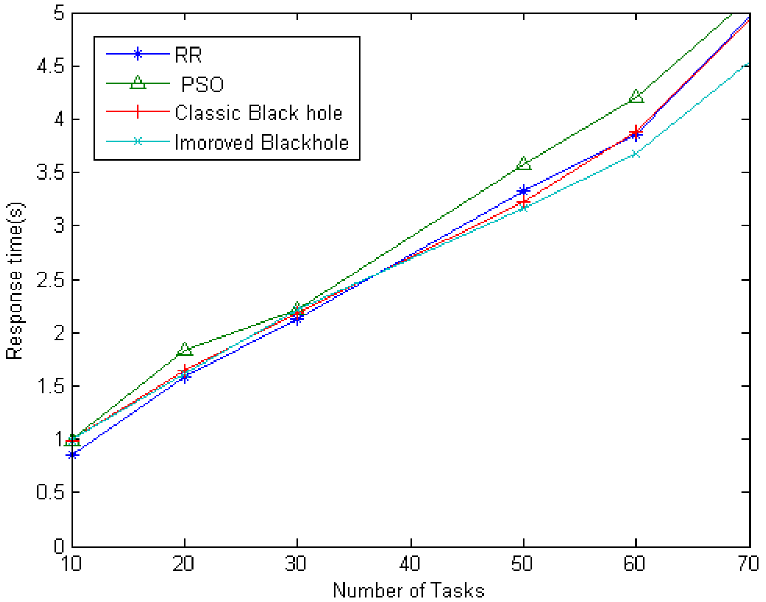

FIGURE 5response time comparison

Comparison of resource utilization among the proposed method, PSO algorithm and RR algorithm is shown in Fig. 6. It is evident that both Modified Black hole-based and PSO-based task scheduling methods results in higher resource utilization comparing to the RR algorithm. Because of its ability of balancing the load of virtual machines, our proposed method has better performance in comparison with the classic PSO algorithm.

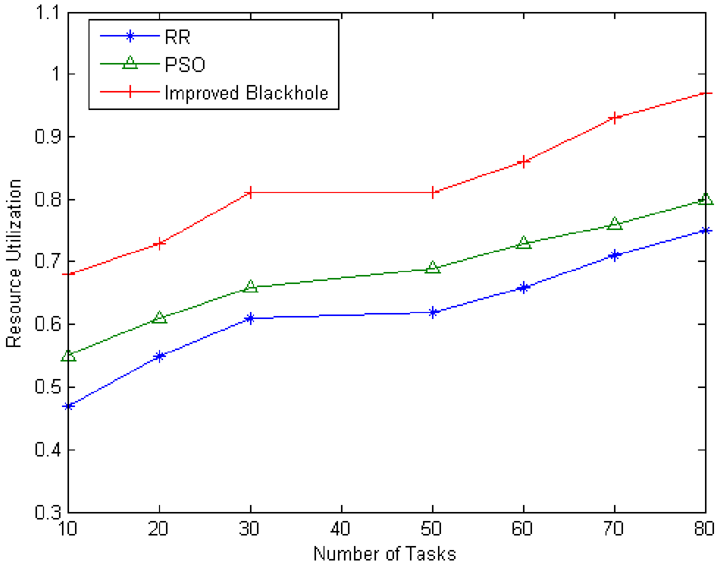

FIGURE 6 Resource utilization comparison

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

- **Test setup 2**

To Analyze performance of the proposed method under real workloads, we have conducted another experiment with a new test configuration. Four hosts, with technical specifications provided in table. 4, are supposed to work in a cloud data center. 40 homogeneous virtual machines are supposed to be running on these hosts. Hardware specification of virtual machines are presented in table. 5.

TABLE 4
Hosts' technical specifications

CPU	Number of cores	Processing speed (Mips)	Ram (MB)	Hard (MB)	Bandwidth (Mbps)
Core_2_Extreme_X6800	2	27079	20480	1048576	102400
Core_i7_Extreme_Edition_3960X	6	177730	1024	1048576	102400
Core_i7_Extreme_Edition_980X	6	147600	20480	1048576	102400
Core_i7_875K	4	92100	20480	1048576	102400

TABLE 5
Virtual machines' technical specification

CPU	Number of cores	Processing speed (Mips)	Ram (MB)	Hard (MB)	Bandwidth (Mbps)
Core_i4_Extreme_Edition	1	9726	512	10240	1024

We have used a workload which is logged by NASA Ames Research Center from October to December in 1993 [15]. It contains 42240 jobs. Each job is converted to a Cloudlet regarding to its completion time and processing rate of existing processors. Each Cloudlet is a task that can be used in CloudSIM simulator.

Fig. 7 shows makespan comparison of four algorithms. With fixed number of virtual machines (40 virtual machines), number of jobs is increased from 300 to 2500. It is evident that our proposed method works efficiently under real workloads and outperforms other three algorithms in terms of makespan duration.

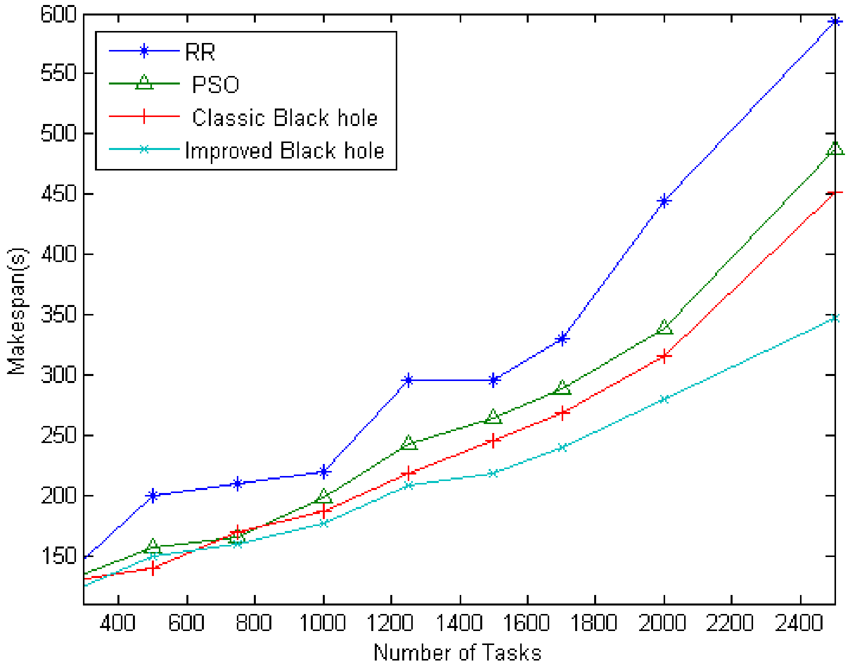

FIGURE 7 Makespan time comparison with real workload

In addition, The Simulation results illustrate that our proposed method increases resource utilization in the entire system, as well as decrease makespan, and increase response time. Moreover, the proposed method also is more efficient from computational complexity point of view and has simple implementation in comparison with other heuristic algorithms.

5. CONCLUSION AND FUTURE WORK

In this paper, a method of scheduling based on black hole algorithm was presented. Using this algorithm and choosing a suitable fitness function, increment of the resources utilization and the reduction of the makespan are feasible. Then proposed method with three algorithms of round robin algorithm as well as algorithms of PSO [13] and base black hole as one of the heuristic algorithms suitable for the issue of scheduling were compared. The simulation results show that in spite of the computational simplicity, the modified black hole algorithm has a good optimization in terms of makespan and resource utilization. Our method is generic and scalable as it can be deployed in data centers with any number of tasks and resources by increasing task-resource array dimension. Furthermore, our method is applicable for any cloud environments with independent and non-preemptive tasks. In the future we plan to expand our method for workflow applications and taking other QoS criteria like fault tolerance capability and cost reduction into account.

REFERENCES

[1] K. A. Nuaimi, N. Mohamed, M. A. Nuaimi, and J. Al-Jaroodi, "A survey of load balancing in cloud computing: Challenges and algorithms," in Network Cloud

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

Computing and Applications (NCCA), 2012 Second Symposium on, 2012, pp. 137-142.

[2] Abdolreza Hatamlou, "Black hole: A new heuristic optimization approach for data clustering", Information Sciences , journal homepage: www.elsevier.com/locate/ins ,2013. pp.175–184.

[3] L. Wang and L. Ai, "Task Scheduling Policy Based on Ant Colony Optimization in Cloud Computing Environment," in LISS 2012, ed: Springer, 2013, pp. 953-957.

[4] S. K. Chaharsooghi and A. H. M .Kermani, "An effective ant colony optimization algorithm (ACO) for multi-objective resource allocation problem (MORAP)," Applied Mathematics and Computation, vol. 200, pp. 167-177, 2008.

[5] Y. Fang, F. Wang, and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing," in Web Information Systems and Mining, ed: Springer, 2010, pp. 271-277.

[6] P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Applied Soft Computing, vol. 13, pp.2013 ,2303-2292 .

[7] X. Nan, Y. He, and L. Guan, "Optimal resource allocation for multimedia application providers in multi-site cloud," in Circuits and Systems (ISCAS), 2013 IEEE International Symposium on, 2013, pp. 449-452.

[8] G. Kaur and E. S. Sharma, "Optimized Utilization of Resources Using Improved Particle Swarm Optimization Based Task Scheduling Algorithms in Cloud Computing," International Journal of Emerging Technology and Advanced Engineering, vol. 4, June 2014 2014.

[9] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, 2010, pp. 400-407.

[10] T. Kokilavani and D. D. G. Amalarethinam, "Load balanced min-min algorithm for static meta-task scheduling in grid computing," International Journal of Computer Applications, vol. 20, pp. 43-49, 2011.

[11] S. A. M. Solmaz Abdi, Saeed Sharifian, "Task Scheduling using Modified PSO Algorithm in Cloud Computing Environment," presented at the International Conference on Machine Learning, Electrical and Mechanical Engineering (ICMLEME'2014), Dubai (UAE), 2014.

[12] S. Zhan and H. Huo, "Improved PSO-based task scheduling algorithm in cloud computing," Journal of Information & Computational Science, vol. 9, pp. 3821-3829, 2012.

[13] K. KRISHNASAMY, "TASK SCHEDULING ALGORITHM BASED ON HYBRID PARTICLE SWARM OPTIMIZATION IN CLOUD COMPUTING ENVIRONMENT," Journal of Theoretical & Applied Information Technology, vol. 54, 2013.

[14] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for the modeling and simulation of cloud resource management and application provisioning techniques," Journal of Software: Practice and Experience, 2010.

[15] D. G. Feitelson and B. Nitzberg, "Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860," in Job Scheduling Strategies for Parallel Processing, 1995, pp. 337-360