© 2018 IJSRSET | Volume 4 | Issue 6 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section : Engineering and Technology

Task Scheduling using Adaptive PSO Algorithm in Cloud Computing Environment

[1]B. SivaRama Krishna, [2]Dr. T. V. Rao
[1]Research Scholar, Department of Computer Science and Engineering, ANU, India
[2]HoD, Department of Computer Science and Engineering, PVPSIT, India

ABSTRACT

Task scheduling problem is one of the most important steps in using cloud computing environment capabilities. Different experiments show that although having an optimum solution is almost impossible but having a sub-optimal solution using heuristic algorithms seems possible. In this paper three heuristic approaches for task scheduling on cloud environment have been compared with each other. These approaches are PSO algorithm, ACO and adaptive PSO algorithm for efficient task scheduling. In all these three algorithms the goal is to generate an optimal schedule in order to minimize completion time of task execution.

Keywords— Cloud environment, ACO, PSO, Task scheduling.

I. INTRODUCTION

Cloud computing has emerged as one of the most progressive, proficient and accommodating technical platform for users of all capacities. It is accepted widely as a utility service where numerous servers are connected to Internet. The cloud users can access data, process services, store data, retrieve data for domestic as well as commercial purposes without owning datacenter, software, hardware and server by paying for its usage from any geographical region having Internet. Mainly three types of services are provided by the cloud. First is Infrastructure as a Service (IaaS), which provides cloud users the infrastructure for various purposes like the storage system and computation resources. Second is Platform as a Service (PaaS), which provides the platform to the clients so that they can make their applications on this platform. Third is Software as a Service (SaaS), which provides the software to the users; so users don't need to install the software on their own machines and they can use the software directly from the cloud. Cloud providers exploit virtualization technology and supply their clients with computing resources in the form of virtual machines (VMs). Service providers, on the other hand, benefit from these VMs to provide users with application level services. To assign users' tasks to VMs, reduce the response time, provide promising quality of service (QoS), and make the most of the resource, service providers exploit task scheduling techniques. Therefore, the task scheduling algorithm is one of the core elements of each cloud infrastructure. Task scheduling is one of the most important and critical problems in cloud computing and many researches have tried to find an optimal solution for scheduling tasks on existing resources in cloud environment.

1.1 Scheduling Problem

Scheduling problem is how to allocate tasks with limited resources to achieve some pre-set goals. The goal of cloud computing scheduling is to achieve the optimal scheduling submitted by the user. In order to improve the overall throughput of cloud computing systems with Specific objectives include the optimal makespan, quality of service,(QoS), load balance, economic principles and so on.

1.1.1. Optimal Makespan

Makespan is a very important and common goal in task scheduling. Users usually hope that their tasks can be completed as soon as possible. Optimal makespan is the common goal of both cloud provider and clients.

1.1.2. Quality of Service (QoS)

Scheduling system must guarantee the QoS specified by the users. On the one side, it needs to improve the efficiency of resources based on application characteristics in order to ensure the efficiency and accuracy of customers. On the other side, it should select and redirect resources dynamically based on users' status changes to meet the user's economy and satisfaction. So the goal is not only to protect users but also helpful for the long-term sustainable development of cloud computing.

1.1.3. Load balancing

Since the number of computers in the cloud computing platform is very large. In additional, the complex composition and different heterogeneous cloud computing platform make load balancing in current could challenging.

1.1.4. Economic principles

Economic is a key factoring in scheduling of cloud computing because of ultra large scale and pay-per-use business model. Market driven cloud users and providers can have mutual benefits from an efficient scheduling system.

1.1.5. Throughput of the system

Mainly for cloud computing systems, throughput is a measure of system task scheduling optimizing performance, and it is a target, which has to consider in business model development. Increase throughput for users and cloud providers would be benefit for them both.

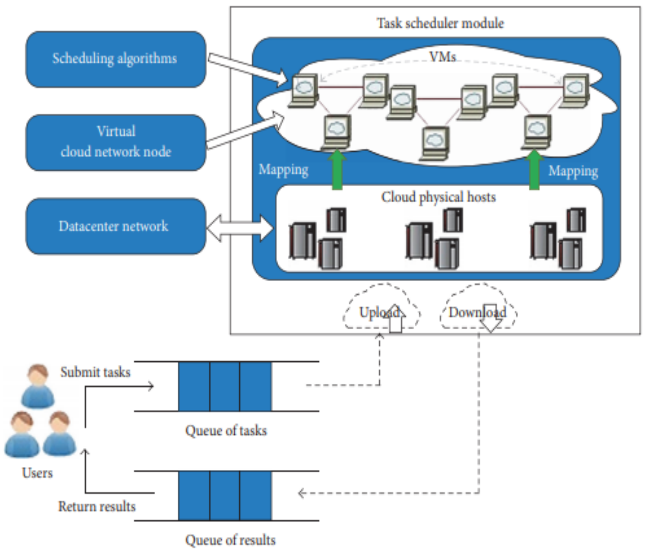


Fig.1. Cloud scheduling model

1.2. Scheduling Problem Formulation

In the following, I adopt the general model and notation used by existing works on PSO-based scheduling. A workflow is usually represented by a Directed Acyclic Graph (DAG), and denoted by $G = (V, E)$. The set of nodes $V = \{T_{1,}, ..., T_n\}$ represents the tasks in the workflow applications, and n is the total number of tasks in the workflow. The arcs $E = \{d_{ij}\}, 1 \le i, j \le n$ denotes the data dependencies among the tasks. An arc, $dij = (T_i, T_j) \in E$ implies that T_i transfers data to T_j . In this relationship, T_i is the parent task of T_j , and T_j is the child of T_i . The child task cannot be executed without receiving data transferred from all of its parents.

Suppose there are a total of m resources in the cloud environment. The resources can be denoted as $R = \{R_1, R_2, ..., R_m\}$ .All the resources are interconnected with each other so that they can transfer data among each other. The scheduling problem is to find an optimal mapping M between tasks and resources according to some optimization objective. As mentioned before, cost is a common objective that is more concerned by user; makespan is another objective that is critical for scheduling. Let

$Makespan_{total}(M)$ denote the makespan of the workflow with respect to the mapping *M*.

$$Makespan_{total}(M) = finish\ time\ of\ the\ last\ task - start\ time\ of\ the\ first\ task$$

The makespan of a workflow is the time duration from the process of first task till finishing all tasks. Since a workflow consists of interdependent tasks, both execution time and transfer time need to be considered.

Let $Cost_{exec}(R_i)$ and $Cost_{trans}(R_i)$ be the execution and transfer costs of resource R_i , respectively. $Cost_{total}(R_i)$ denotes the total cost of resource R_i

$$Cost_{total}(R_i) = Cost_{exec}(R_i) + Cost_{trans}(R_i)\ 1 \le i \le m$$

Let $Cost_{total}(M)$ denote the total cost of processing workflow w.r.t the mapping M:

$$Cost_{total}(M) = \sum_{i=1}^{m} Cost_{total}(R_i)$$

For the objective of minimizing the cost while balancing the load, the fitness function is given as:

$$Fitness\ function_1 = Max(Cost_{total}(R_i)) \quad 1 \le i \le m$$

The objective is to minimize $Fitness\ function_1$. The reason for not using the total cost of all the resources is to prevent from mapping all the tasks to a single, least-cost resource. For the objective of optimizing makespan, the fitness function can be defined as:

$$Fitness\ function_2 = Makespan_{total}(M)$$

The objective is to minimize $Fitness\ function_2$

The objective of minimizing the weighted sum of total cost and makespan; the fitness function can then be defined as:

$$Fitness\ function_3 = \alpha \cdot Cost_{total}(M) + (1-\alpha) Makespan_{total}(M), \quad 0 \le \alpha < 1$$

where α is the weight given to the total cost and $(1-\alpha)$ is the weight given to makespan. This fitness function can be easily tuned by changing the value α to satisfy the various QoS requirements including budget constraints.

II. Task scheduling using Adaptive Task Scheduling PSO Algorithm

2.1. Basic description of PSO

PSO is a swarm intelligence meta-heuristic inspired by the group behavior of animals, for example bird flocks or fish schools. Similarly to genetic algorithms (GAs), it is a population-based method, that is, it represents the state of the algorithm by a population, which is iteratively modified until a termination criterion is satisfied. In PSO algorithms, the population $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ of the feasible solutions is often called a swarm. The feasible solutions $p_1, p_2, ..., p_n$ are called particles. The PSO method views the set $\mathcal{R}_d$ of feasible solutions as a "space" where the particles "move". For solving practical problems, the number of particles is usually chosen between 10 and 50. The purpose particle swarm optimization (PSO) algorithm is to solve an unconstrained minimization problem: find x^* such that $f(x^*) \le f(x)$ for all *d*-dimensional real vectors x . The objective function $f : \mathcal{R}_d \to \mathcal{R}$ is called the fitness function.

2.1.1. Swarm Topology

Each particle i has its neighborhood $\mathcal{N}_i$ (a subset of $\mathcal{P}$). The structure of the neighborhoods is called the swarm topology, which can be represented by a graph. Usual topologies are: fully connected topology and circle topology.

2.1.2. Stopping Rule

The algorithm is terminated after a given number of iterations, or once the fitness values of the particles (or the particles themselves) are close enough in some sense.

2.1.3. The Scheduling System

Figure 1 illustrates an overview of the scheduling system. The system consists of three modules, the first module is the application which represents the set of the cloudlets (tasks). The second module is the Mapping Algorithms (MA) which estimates the expected time for each cloudlet to allocate on each virtual machine, and it is assumed that these values are available to the scheduler. A third module is a virtual machine (VMs) which used to execute the cloudlets. The cloudlets expected time have been stored in a $m \times n$ matrix, where m the number of is virtual machines, and n is the number of cloudlets. Obviously, n/m will generally be greater than 1, with more cloudlets than virtual machines, so that some machines will need to be assigned multiple cloudlets. The Estimated Running Time (ERT) is defined as the time of executing task j on resource r [21]. Each column of the expected running times (ERT) matrix contains the expected running time (ERT) of each cloudlet j on machine i.

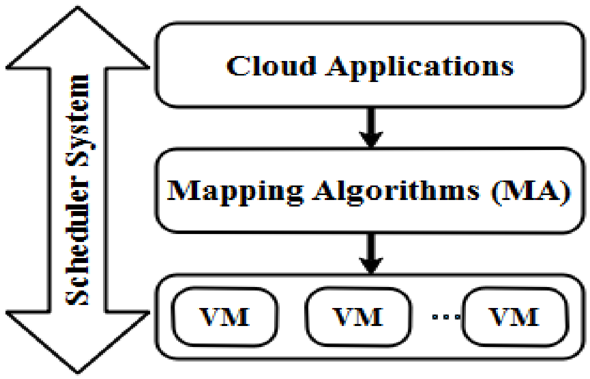


Fig.2.Scheduling System

2.2. The Scheduling System

The main objective of allocating tasks on virtual machines is to reduce the makespan. The makespan of a task is defined as the overall task completion time. We denote completion time of task $\mathcal{T}_i$ on $\mathcal{VM}_j$ as $\mathcal{CT}_{ij}$. Hence, the makespan is defined using the following equation

$$\mathcal{CT}_{\max}[i, j] \mid i \in \mathcal{T}, i = 1, 2, .., n \text{ and } j \in \mathcal{VM}, j = 1,2,.., m \qquad (1)$$

Where $\mathcal{CT}_{\max}[i, j]$ is the maximum completion time of task i on a $\mathcal{VM}_j$, and n, m are the number of tasks and virtual machines respectively.

Let $\mathcal{VM} = \mathcal{VM}_1, \mathcal{VM}_2, ..., \mathcal{VM}_m$ be the number of m virtual machines that must be processed n tasks represented by the group $\mathcal{T} = \mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_n$. The virtual machines are parallel and independent, and the schedule allocates independent tasks to these VMs. Also, the Processing a task on a virtual machine cannot interrupt (i.e.) Non-preemption. We denote end time of a task $\mathcal{T}_i$ by $\mathcal{CT}_j$. The aim of the proposed algorithms is to reduce the makespan which can be denoted as $\mathcal{CT}_{\max}$. The run time of each task for each virtual machine must be calculated for the purpose of scheduling, if the processing speed of a virtual machine $\mathrm{VM_j}$ is $\mathrm{PS_j}$, then the processing time for task $\mathcal{P}_i$ can be calculated by equation.

$$\mathcal{T}_{ij} = C_i / \mathcal{P}S_j \qquad (2)$$

Where $\mathcal{P}_{ij}$ is the processing time of task $\mathcal{P}_i$ by virtual machine $\mathcal{VM}_j$ and CI is the computational complexity of the task $\mathcal{P}_i$.The processing time $\mathcal{P}_{ij}$ of each task $\mathcal{P}_i$ on $\mathcal{VM}_j$ are stored in the runtime matrix. The processing time of each task in the virtual machines can be calculated by equation

$$\mathcal{P}_{ij} = \sum_{i=1}^{n} \mathcal{P}_{ij} \qquad (3)$$

According to (1), (2) and (3), the task scheduling algorithm should satisfy the following equation

$$\sum_{i=1}^{n} \mathcal{P}_{ij} \le \mathcal{CT}_{\max} \qquad (4)$$

By considering the load balancing, the tasks will be transferred from one VM to other to reduce $\mathcal{CT}_{\max}$, as well as, response time. The processing time of a task varies from one VM to another based on the speed of the virtual machines. In case of transferring, the completion time of a task may vary because of load balancing, optimally. The main objective of the

Adaptive Task Scheduling PSO Algorithm is that the tasks should be allocated on the virtual machine in order to minimize the makespan and maximize the resource utilization.

Adaptive Task Scheduling PSO Algorithm

Initialization: Initialize position vector and velocity vector of each particle.

Conversion to discrete vector: Convert the continuous position vector to discrete vector.

Fitness: Calculate the fitness value of each particle using fitness function.

Calculating $\mathcal{P}_{best}$ **:** Each particle's $\mathcal{P}_{best}$ is assigned its best position value till now. If particle's current fitness value is better than particle's $\mathcal{P}_{best}$, then replace $\mathcal{P}_{best}$ with current position value.

Calculating $\mathcal{G}_{best}$ **:** Select the particle with best fitness value from all particles as $\mathcal{G}_{best}$.

Updation: Update each particle's position vector and velocity vector using following equations:

$$V_i + 1 = \Psi V_i + c_1 \text{rand}_1 * (\mathcal{P}_{best} - x_i) + c_2 \text{rand}_2 * (\mathcal{G}_{best} - x_i)$$

$$X_i + 1 = X_i + V_i + 1$$

Where $\Psi = \text{inertia}$, $c_1, c_2 = \text{acceleration coefficients}$, rand1, rand2=uniformly distributed random numbers and $\varepsilon[0,1]$, $\mathcal{P}_{best}$ =best position of each particle ,

$\mathcal{G}_{best}$ = best position of entire particles in a population , i=iteration .

Repeat steps 2 to 6 until stopping condition is met. Stopping condition may be the maximum number of Iterations or no change in fitness value of particles for consecutive iterations.

Output: Print best particle as the final solution.

III. Results and Analysis

This strategy is performed on the cloudsim 3.0.3 with Eclipse Jee Oxygen IDE on windows 10 platform with core i5 processor with 8 GB RAM and 2 GB Radeon graphics card. The language used by cloudsim is JAVA which provides simulating environment of cloud. The algorithm is implemented by considering the parameters like average round trip time average cost, average execution time, average make span. The PSO used in the strategy is with mutation and provides better results when compared with other genetic algorithm and other modified version of PSO. The APSO is implemented and compared with the longest VM longest cloudlet algorithm, genetic algorithm and standard PSO then it provides the optimized results. This strategy is used with increasing number of tasks i.e. 100,200 upto1000. APSO is performing better than other two as we increase the number of tasks.

3.1. Makespan Comparison of Fixed VMs

In this section, simulation is conducted to evaluate the efficiency of the proposed scheduling approach to the optimized solution. The results of the proposed APSO algorithm is compared with other heuristics algorithms such as PSO and ACO from performance parameters like makespan and throughput. Simulation outcomes show that our proposed algorithm outperforms than other heuristics algorithm.

Table.1.Makespan Comparison of Fix VM

S.No	No. of Task	VM	APSO	PSO	ACO
1	100	50	17.04	25.32	28.55
2	200	50	52.01	109.00	117.15
3	300	50	104.96	176.19	198.90
4	400	50	175.95	271.50	279.56
5	500	50	264.96	561.50	454.29

6	600	50	371.98	579.70	667.71
7	700	50	497.03	945.23	955.52
8	800	50	647.99	834.54	1103.58
9	900	50	818.99	1032.28	1167.71
10	1000	50	1010.00	1554.21	1625.86

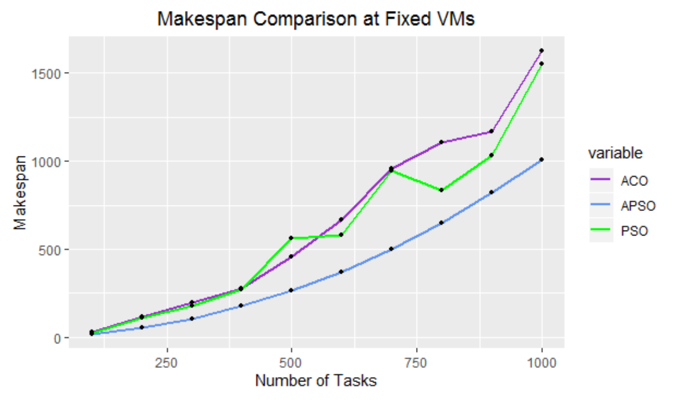


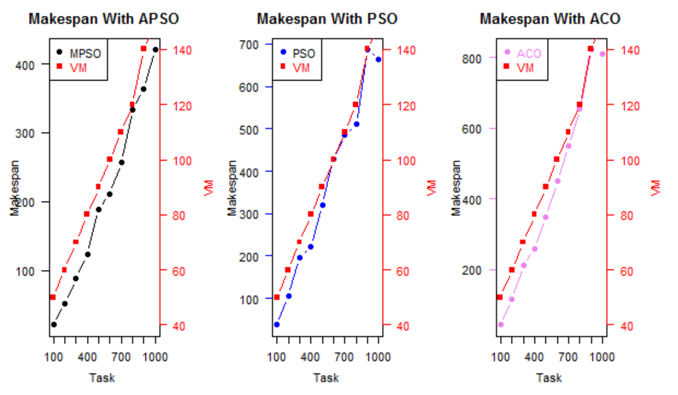
Fig.3.Makespan Comparison of Fix VM

3.2. Comparison of Makespan of Variable VMs

The Makespan of proposed APSO algorithm compared by PSO standard and ACO. This test has been implemented more than 50 times using timeshare policy at the independent nature of task, and the result has been presented. The makespan has been compared by varying the numbers of tasks 100 to 1000 while keeping a fixed number of VMs 50. Further, the results have conducted at varying number of tasks from 100 to 1000 and a varying number of VMs from 40 to 140. The experiment result has been shown in Table 2 and 3 and depicted in Fig.3 and 4. The Makespan produced by APSO algorithm is improved compared with makespan produced by PSO standard and ACO.

Table. 2. Makespan Comparison of variable VMs

S.No	No. of Task	VM	APSO	PSO	ACO
1	100	40	20.71	37.32	43.72
2	200	50	51.99	105.50	116.51
3	300	60	88.15	197.32	213.11
4	400	70	122.60	222.62	258.22
5	500	80	189.30	319.05	347.72
6	600	90	210.30	428.01	449.02
7	700	100	257.38	484.34	549.54
8	800	110	333.30	510.88	655.93
9	900	120	362.80	687.09	822.36
10	1000	140	421.09	663.96	809.22



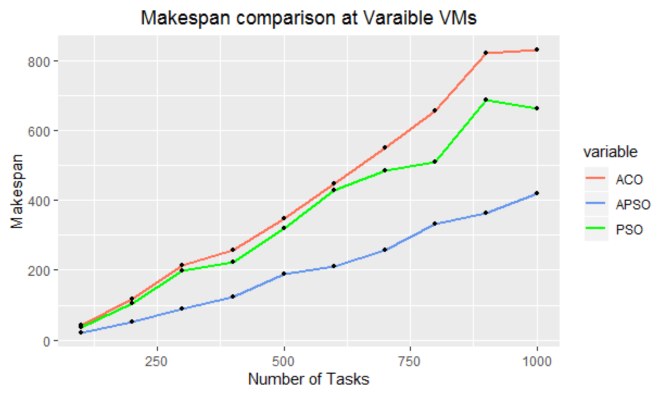


Fig.4. Makespan Comparison with variable VMs

3.3. Comparison of Throughput

The comparison of throughput of proposed APSO, ACO and PSO is illustrated in Table 3 and depicted in Fig.5. The performance parameter is computed for analysing the maximum throughput. Table 3 and Fig. 5 shows that the performance of the APSO algorithm improved even when numbers of tasks are increased from 100 to 1000 while fixed number of VMs 50. Table 4 and Fig. 6 shows the throughput of the proposed task scheduling algorithm is also improved when varying the number of tasks from 100 to 1000 and VMs from 40 to 140. The throughput of the proposed algorithm is much improved when compared to PSO standard and ACO algorithm.

Table.3. Throughput Comparison of variable VMs

S.No	No. of Task	VM	APSO	PSO	ACO
1	100	40	4.82	2.67	2.87
2	200	50	3.90	1.89	1.71
3	300	60	3.44	1.52	1.40
4	400	70	3.26	1.79	1.54
5	500	80	2.64	1.56	1.43
6	600	90	2.85	1.40	1.33
7	700	100	2.75	1.44	1.27
8	800	110	2.40	1.56	1.21
9	900	120	2.48	1.30	1.09
10	1000	140	2.37	1.50	1.23

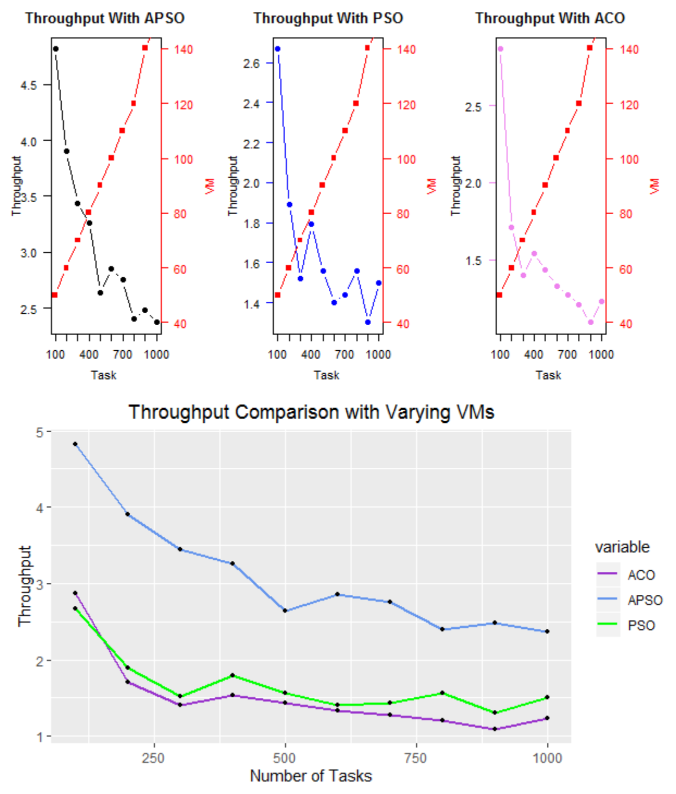
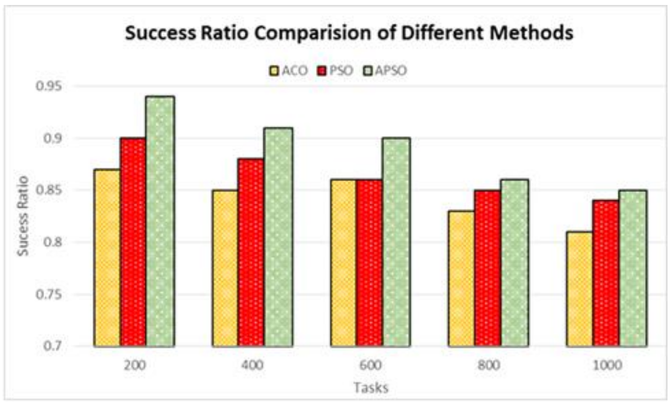


Fig.5.Throughput Comparison of variable VMs

Table.4.Success Ratio Comparison with Tasks

S.No	Task-Resource Scheduling Algorithm	No. of Task	Success Ratio
1	ACO	200	0.87
	PSO		0.9
	APSO		0.94
2	ACO	400	0.85
	PSO		0.88
	APSO		0.91
3	ACO	600	0.86
	PSO		0.86
	APSO		0.9
4	ACO	800	0.83
	PSO		0.85
	APSO		0.86
5	ACO	1000	0.81
	PSO		0.84
	APSO		0.85



IV. Conclusion

In this paper the problem of task scheduling in cloud computing environment is evaluated. PSO algorithm and ACO are most famous algorithms for scheduling tasks in distributed systems. In order to improve the performance of standard PSO algorithm the Adaptive PSO algorithm is suggested, in which objective function is modified in the standard PSO algorithm for generating initial population in order to minimize makespan. Our experiments depicts that even if both ACO and PSO algorithm show acceptable results, it can be said that by and large PSO algorithm shows better results than ACO but modified PSO algorithm outperforms these two algorithms from minimizing makespan point of view. This algorithm can be used in cloud computing environment for efficient scheduling of tasks on existing resources, so that completion time of tasks become minimized. PSO based scheduling algorithm in cloud computing.

V. REFERENCES

[1]. Cloud computing. Peng Liu:cloud computing definition and characteristics http://www.chinacloud.cn/.2009-2-25.

[2]. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, “Cloud Computing and Emerging IT Platforms”, Vision, Hype, and Reality for Delivering Computing as the 5th Utility, Future Generation Computer Systems 25(6), 599–616 (2009). http://dx.doi.org/10.1016/j.future.2008.12.001

[3]. P. Kumar, A. Verma, “Independent Task Scheduling in Cloud Computing by Improved Genetic Algorithm”, International Journal of Advanced Research in Computer Science and Software Engineering,Vol2, Issue 5, May 2016.

[4]. Z. Yingfeng, L. Yulin, “Grid Computing Resource Management Scheduler Based on Evolution Algorithmj]”, Computer Engineering Conference, 2003, 29(15):1102175.

[5]. P. Roy, M. Mejbah, N. Das. “Heuristic Based Task Scheduling in Multiprocessor Systems with Genetic Algorithm by choosing the eligible processor”, International Journal of Distributed and Parallel Systems (IJDPS), Vol3, No.4, July 2017.

[6]. Abraham, R. Buyya, and B. Nath.” Nature’s heuristics for scheduling jobs on computational Grids”, 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), India, 2000.

[7]. H. Yin, H. Wu, J. Zhou, “An Improved Genetic Algorithm with Limited It rat ion for Grid Scheduling”, IEEE Sixth International Conference on Grid and Cooperative Computing, GCC 2007, Los Alamitos, CA, pp. 221-227, 20013.

[8]. R. Verma , S. Dhingra, “Genetic Algorithm for Multiprocessor Task Scheduling”, IJCSMS International Journal of Computer Science and Management Studies, Vol.1, Issue 02, pp. 181-185, 2011

[9]. J. Kennedy, R.C. Eberhart, “Particle swarm optimization”, Proc, IEEE Conf. Neural Netw., vol. IV, IEEE, Piscataway, NJ, 1995,pp.1942-1948.

[10]. L. Zhang, Y. Chen, B. Yang “Task Scheduling Based on PSO Algorithm in Computational Grid”, 2013 Proceedings of the 6th International Conference on Intelligent Systems Design and Applications, vol-2, 16-18 Oct, 2013, Jinan, China.

[11]. T. Chen, B. Zhang, X. Hao, Y. Dai, “Task scheduling in grid based on particle swarm optimization”, The Fifth International Symposium on Parallel and Distributed Computing, ISPDC '06. pp. 238-245, 20015.