A Proposed "Modified Filtering Algorithm" (MFA) for Secured Data by Identifying the Delay and Effective Completion Time of Tasks in Cloud

A. Manimuthu, G. Murugaboopathi

Abstract--- In today's era, cloud computing has played a major role in providing various services and capabilities to a number of researchers around the globe. One of the major problems we face in cloud is to identify the various constraints related with the delay in the Task accomplishment as well as the enhanced approach to execute the task with high throughput. Many researches have shown that although having an optimum solution is almost impossible but having a sub-optimal solution using heuristic algorithms seems possible. In this paper, we propose "Modified Filtering Algorithm" for task scheduling on cloud environment, compared with previously used particle swarm optimization (PSO), heuristic approaches, and modified PSO algorithm for efficient task scheduling. Comparing all these three algorithms, we aim to generate an optimal schedule in order to minimize completion time of task execution.

Keywords--- Cloud environment, Modified Filtering Algorithm (MFA), Heuristic algorithms, PSO, Task scheduling, Quantum Time;

I. INTRODUCTION

In our paper, the calculation of task execution is been done using MFA followed by hereditary calculation and altered PSO calculation for errand planning issue in Cloud figuring condition. The heuristic algorithms have been analyzed and our aim is to show the contrast between each of the previously used approaches and algorithms. We have explored MFA which is called hereditary calculation. Cloud computing is developed from the grid computing, distributed computing and parallel computing. It is a new business computing model and will distribute the tasks on the resources pool which is made up with the large number of computers, storage devices [1]. This paper is take after in which the planning issue is displayed and portrayed. Errand planning is a standout amongst the most vital and basic issues in distributed computing and numerous exploration have been attempted to locate an ideal answer for booking assignments on existing assets in cloud condition. A developmental calculation is utilized to discover a problematic arrangement of the issue in significant way as proposed by R. Buyya[2], the first ever initiation for MFA.

We have adopted a Parallel and appropriated framework comprising of a gathering of interconnected and virtual PCs that are powerfully provisioned, and introduced as at least one brought together registering assets in view of administration level assertions set up through transaction between the specialist co-op and customers [2]. 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 [3]. 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. But the problem of task scheduling has several problems in general as proposed by Z. Yingfeng et. al. [4]. In this paper, we have used MFA in which the number of assumptions are almost non-fuzzy and very less, in fact negligible. Non Fuzzy in the sense that the constants that we have assumed here are coefficients and these coefficients mostly are considered as constants rather than assumptions. MFA, in this paper, can be considered as an evolutionary algorithm that is used to find a sub-optimal solution of the problem in considerable to identify the delay and effective completion time of a given set of tasks in cloud. To reach the solution faster many heuristic based approach were used in the past Roy et. al. [5].

Considering the previous of Z. Yingfeng et. al.; Roy et. al.; Yin et. al.; Verma et. al.; they have used stagnant assumptions like convergence, normalization, recurrence relation, parameters related with stability. We can also see that all the works published by the authors mentioned have given the future resolution in their work as these assumptions have affected the scheduling, increased the complexity of the task accomplishment and most importantly they don't give the full assurance of job execution, rather they have approached in a predictive manner. We also agree that their work has given better results than the previous works as mentioned in the literature survey.

In the context of our paper, we have absolutely outcasted these highly weighted assumptions (convergence, normalization, recurrence relation, parameters related with

Manuscript received February 01, 2019

A. Manimuthu, Research Scholar, Department of Computer Science, Bharathiar University, Coimbatore, Tamil Nadu, India.
Dr.G. Murugaboopathi, Associate Professor, Department of Computer Science and Engineering, Kalasalingam University, Krishnankoil, Tamilnadu, India.

stability), rather we have used coefficients. Using coefficients overcome the scheduling in a very efficient way because it doesn't affect the major characteristics of the scheduling of a particular task. Our experimental results justifies this statement. The assumptions we have used as a coefficient has helped scheduling on cloud environment more accurately than the previously used particle swarm optimization (PSO), heuristic approaches, and modified PSO algorithm for efficient task scheduling.

Abraham et al. [6] have investigated three different nature based heuristic algorithms, which are called genetic algorithm, simulated annealing and Tabu search. Genetic algorithm and simulated annealing are inspired from nature. Genetic algorithm simulates natural selection process. The solutions that we have proposed in this paper are evaluated by fitness function for individual task which have better fitness value generated along with new solutions. Using modified filtering algorithm for task scheduling, the results that we have achieved is far better than Particle Swarm Optimization (PSO) and Modified PSO that have been addressed in Yin et. al. [7] and Verma et. al. [8]. One of the mostly used algorithms is PSO. This algorithm that has developed by Kennedy and Eberhart [9] is one of the evolutionary algorithms which simulates social behavior of flock of birds or groups of fishes toward their desired destination. In [10] a PSO algorithm is introduced by Shih Tang Lo, et al in which the problem of multi-processor resource scheduling is considered. In addition to benefits of heuristic algorithms, such as flexibility and acceptable calculations, PSO has single in kind specifications such as easy implementation and consistent performance. Nowadays PSO algorithm is used in different domains such as neural network training, control system analysis and design, structural optimization and so on. In this paper, PSO algorithm, genetic algorithm and modified PSO algorithm for task scheduling problem in Cloud computing environment are compared with each other. Simulation results show that even if three algorithms show acceptable results, but totally, Modified Filtering Algorithm performance is better than two other algorithms.

In short, identification of various parameters that are related with the delay and effective completion time of a given set of tasks in a particular cloud is the major issue of our work. We have accomplished with a novel filtering approach to compute these parameters. We claim that our work is optimal; the reason as mentioned in the previous paragraph is the use of negligible assumptions. Hence, we propose "Modified Filtering Algorithm" for task scheduling on cloud environment, compared with previously used particle swarm optimization (PSO), heuristic approaches, and modified PSO algorithm for efficient task scheduling. Comparing all these three algorithms, we aim to generate an optimal schedule in order to minimize completion time of task execution.

II. LITERATURE SURVEY

In recent years, the phenomenal advancement in computing speed, development in network technologies and demand for solving large scale problems have led us into the era of high performance distributed computing widely known as Grid computing. The concept of Gridwas proposed by [1, 2]. They introduced Grid as a collection of large number of heterogeneous, distributed, decentralized and dynamic resources. The resources may be distributed over multiple control domains and interconnected through high speed communication network. The purpose of Grid is to provide a high performance computing platform for executing large scale applications, which may contain a number of dependent or independent jobs. Appropriate scheduling of the applications onto the Grid resources plays an important role in achieving the goals of Grid

The servers of a data center account for the largest amount of energy consumed by the data center [3]. Recent works focus on schemes aiming at reducing energy consumption by servers through efficient job scheduling, resource allocation optimization, and virtual machine consolidation [4]. A conservative allocation of resources and jobs in data centers may lead to powering ON a large number of servers, contributing to a large amount of consumed energy [5]. Energy-aware job allocation schemes may be used towards receding the energy consumed by servers [6]. In such a scheme, the traffic distribution and link states of a data-center network are considered for deciding to which servers jobs are allocated. The objective of these schemes is to consolidate network traffic and server load to reduce the fraction of active network equipment, set link speeds to match traffic demand, and turn off non-critical servers.

Cloud-computing data centers may use VMs to consolidate workloads to reduce the number of active servers [9-12,15-19]. To ensure that the service level agreement (SLA) is met, cloud-computing data centers may set up upper limits for resource utilization while placing VMs. This may lead to poor utilization of resources due to the dynamics of data center workloads. A scheme that uses dynamic thresholds was proposed considering the following policies for placing VMs: minimization of VM migration, balancing of resource utilization and SLA violation, and random selection [6]. This scheme achieves energy efficiency by allowing a level of SLA violations. Another approach introduces a power management scheme that implements multiple feedback controllers at the levels of racks, servers, and VMs to improve data center energy efficiency [11]. Although reducing the number of active servers in a data center may improve energy efficiency, over-consolidation may jeopardize the quality of the service (QoS) provided by the data center. Maintaining QoS whilst increasing energy efficiency is critical for the economical sustainability of a data center [10,15-17]. Minimizing the number of virtual-machine migrations may be employed to maintain service guarantees and to reduce the energy consumption of cloud-computing data centers [16].

In a more direct approach, jobs may be mapped to the existing computing resources to ensure that the satisfaction of the required QoS of different applications [19]. However, these methods require global knowledge of the state of the data center and, in turn, a fast central controller to perform

timely decisions for the dynamic data center networks.

III. SCHEDULING PROBLEM MODELING USINGMFA

Our first assumption considers that each task submitted by the user is independent of other tasks. Here, we assume that suppose we have n tasks of which we know of execution time on each processing machine is known and they should be processed on m computational resources and our goal is to minimize completing execution time and optimizing resource utilization.

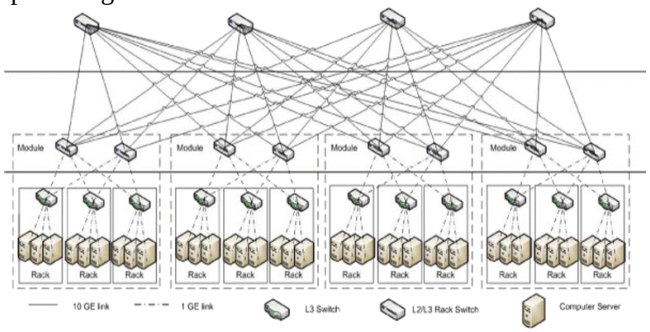

Figure 1: Model of Cloud Computing data center

Referring to the figure 1 above, if the number of tasks is less than resources, tasks are distributed based on first-come-first-serve (FCFS) scheduling algorithm. Otherwise the task distribution is performed based on scheduling algorithms such as Shortest Job first or Round Robin Scheduling et cetera according to the process. In this paper we assume that number of tasks is more than number of resources, so one task cannot be assigned to different resources, i.e. we assume non-preemptive scheduling. To formulate the problem, set of tasks is defined as T = {1,2, ,3,..., n} are 'n' independent jobs and R = {1,2,3,...,m} is set of 'm' computational resources. We know the execution time of process i on computational node j is known, and it is equal to job Run Time ( ). Our goal is to find a matrix M such that if a process i is executed using any resource j then product of and any element of matrix M is 1 otherwise it is 0 and it will minimize the total cost of completion time of task execution on computational resources. In this part two fundamental conditions are considered:

$$\mathcal{Z}_t^n X_{i,j} = 1 \ \forall j \in T (1)$$

$$X_{i,j} \in \{0,1\} \forall i \in R, j \in T \quad (2)$$

- The equation 1 and 2 depict the the filter output $X_{i,j}$ where T being the universe of discloser
- Also, R depict the error occurred (iff) in all 10 iterations
- Most significantly the value ranges is i between 0 to 1

Makespan is equal to the execution of the last task for which the processing resource is finished. First limitation ensures that each task is assigned to only one processing resource.

A. Task scheduling using MFA algorithm: MFA algorithm is one of the most famous evolutionary approaches for optimization which is inspired by the natural selection process. In this approach, characteristics of potential tasks are improved by various defined operators in MFA algorithm. MFA algorithm behaves like heuristic methods. MFA is an attractive computational models for simulating natural evolution for solving delays in accomplishing jobs using many resources. MFA algorithm consists of number of individuals and each one of them is a potential solution for task execution. Using MFA operators, we can change these processes across many generations / populations to find an optimal or suboptimal solution for the same problem. Using evolution theory, only the fittest solutions in the search space have a chance to survive for the next generation and creating offspring of that particular task. The most important aspects of MFA algorithm are:

a) Define the objective function
b) Define and implement the MFA form of the problem,
c) Define and implement the MFA operators.

In mapping between PSO, modified PSO with that of MFA algorithm, individual jobs are now the solutions to the same problem, where population represents the set of solution, fitness demonstrates quality of solutions, and process is a part of representation of the solution. Different steps for MFA algorithm are:

1. **Initialization:** -An initial population is generated randomly. Each individual represents a potential solution of the problem.
2. **Fitness function:-** After creating the initial task, we define the fitness function to evaluate the goodness of individuals. Fitness function is different for each optimization problem as the goals differ. In this problem, we use Makespan as fitness function.
3. **Selection and Elitism:-** Elitism selects the best individuals to go to the next generation without change and selection refers to selecting individuals corresponding to better fitness values for creating new offsprings. For this problem, we randomize the processes for selection.
4. **Crossover: -** Here, the preceding tasks are virtually generated using selected individuals. The most common way for doing this is to choose any two particular tasks, and compare their parameters (For e.g. completion time). We use crossover at such points while solving the problem.
5. **Mutation:** - The second way is that MFA algorithm uses for searching is cost-surface for mutation.

B. Task scheduling using PSO algorithm

B.1. PSO algorithm (particle swarm optimization)

PSO is one of the newest heuristic optimization algorithms proposed by Kennedy and Eberhart [9]. Observing the behavior of a flock of birds or group of fishes in order to get to a desired destination is often used to simulate PSO. We consider a population of particles as the initial space.

Each particle in the population represents a potential solution and corresponds to individuals in the evolutionary algorithms. First, we initialize a population randomly where the position of each particle represents a potential solution in the problem space.

For each particle a position vector is defined which is updated over time ( ), moreover there is a velocity vector ( ) which is used to update the position vector, consistently and leads to movement in search space. Based on proposed algorithm by Kennedy and Eberhart [9] the formula that exists for updating position vector is:

The update rule for velocity vector is given as:

$$v_{t+1} = w * v_t + \omega_1\alpha_1(p_t - x_t) + \omega_2\alpha_2(g_t - x_t) \quad (3)$$

Then, the position update rule for the solution search space is given as:

$$x_{t+1} = x_t + v_{t+1} \quad (4)$$

In equation (3), α_1 and α_2 are constant numbers, and are numbers with normal distribution in the range of [0, 1]. Velocity vector changes in range of [0,1]. In each generation, we evaluate the fitness functionon position vector of each particle and fitness of each particle is evaluated. Two defined parameters in equation 3 and which play very important roles in the process of PSO. represents the best local position that any particle has experienced since the process has started and represents the overall best position among all particles from beginning of the algorithm. Using velocity vector that is updated regularly, particle is able to search around its own best position ( ) and best global position ( ). is velocity of a particle in time t and represents the distance that a particle should travel. and are acceleration coefficients. w is inertia weight. Bigger w searches in the new spaces but smaller amounts of w searches in the existing space. If we want to balance between local and global search, inertia weight and acceleration coefficient should be opted for appropriately. In the velocity equation, new velocity for a particle is updated based on its previous velocity and its distance from its own best position and the best position among all particles, after a particle moves toward its new position according to above equation. Evaluation of each particle at a given time is done based on proportion function that is defined formerly. This process repeated until termination criterion of algorithm is satisfied or a particular number of iterations.

We propose a modification in the existing MFA algorithm to improve the performance in terms of delay in task injected to and from cloud. The authors have tried to introduce a variance factor in the MFA algorithm, the performance of the algorithm can be improved by adding**ω**'. The modified equation is illustrated in equation 5 and 6 in the following section.

- The ω represents if the delay is occurred for the particular task within 10^{th} iterations and
- τ is an acknowledgement for accomplishment of the task.

The proposed modification was simulated and tested using Cloud-Sim for delay calculation and minimizing the same using *"port knocking"*. The simulation results are illustrated in the following sections.

B.2 MFA algorithm for Task Scheduling

At first we perform an accurate mapping between tasks of PSO and MFA problem solutions. The goal is to improve resource utilization and minimize completion time of tasks. PSO is able to solve many optimization problems in different areas. If we suppose that there are n tasks and we aim to distribute them on m processing machines then tasks should be defined in form of n×m matrices. In this position matrix, m is number of available nodes at scheduling time and n is number of jobs. Position vector of each task has two main characteristics:

- All elements of position matrix are 0 or 1.
- In some of the vector of position matrix (T2, T3, T6, T7 and T8), there is only one 1 and other elements are zero.

Table 1: A typical events for 9 independent tasks and 3 processing machine

	T1	T2	T3	T4	T5	T6	T7	T8	T9
M1	1	0	0	1	0	1	0	1	1
M2	1	0	1	0	1	0	1	0	1
M3	0	1	0	1	1	0	0	0	0

In this matrix, in each column, each element 1 depicts the machine that executes the task. The second condition ensures that each task should be executed on only one machine. For example in the above position vector, T1 is executed on machine 1 and machine 2, T2 is executed on machine 3 only, T3 on machine 2 and so on. This clearly indicates that T1 has some error.

Velocity matrix dimensions are exactly similar to position matrix i.e. $m \times n$. And ranges for its elements are [- $V_{max,}$ + V_{max}]. So for each element in the population velocity matrix is as (5) (V_k is velocity matrix of k^{th} task):

$$V_k^{(i,j)} \in [-v_{max}, +v_{max}] \; \forall \; (i,j) \quad (5)$$

where

$$i \in \{1,2,3,\ldots..m\}$$
$$j \in \{1,2,3,\ldots..n\}$$

Similar to velocity matrix, P_k and G_k are matrices with 0 and 1 elements. $m \times n$ matrices with 0 and 1 elements.

P_k matrix represents a matrix that shows the best position of task from beginning of the algorithm and G_k owns the best position between all tasks from the beginning of the algorithm. In each iteration of the algorithm P_k and G_k are updated based on related formulas, which are explained in the previous sections. In each step of the algorithm tasks are evaluated by fitness function of the algorithm. If task's fitness is better than P_k, then P_k is replaced by X_k. In order to update G_k we use P_k of all tasks. If there is P_k that is better than G_k, current G_k is replaced by P_k.

In order to update position vector of tasks we use velocity matrix of that task:

$$V_k^{t+1}(i,j) = w * v_k^t \; (i,j) + \omega_1\alpha_1(P_k^t(i,j) - X_k^t(i,j)) + \omega_2\alpha_2(G_k^t(i,j) - X_k^t(i,j)) \quad (6)$$

To update position vector of tasks based on in each column we compare values of all rows in that column, the element that has biggest value, its corresponding element in position vector turns to 1 and other elements in that column become 0.

Fitness function is defined based on our objective. In this paper we look for minimizing Makespan, i.e. the time that execution of last task is completed. So for each machine we sum job run time of all tasks that are assigned to that machine and the biggest number is said to be fitness of that

task.

We repeat this process until stop criterion of algorithm is satisfied which in our algorithm it is equal to number of iterations. Flowchart of MFA algorithm is as Fig. 2.

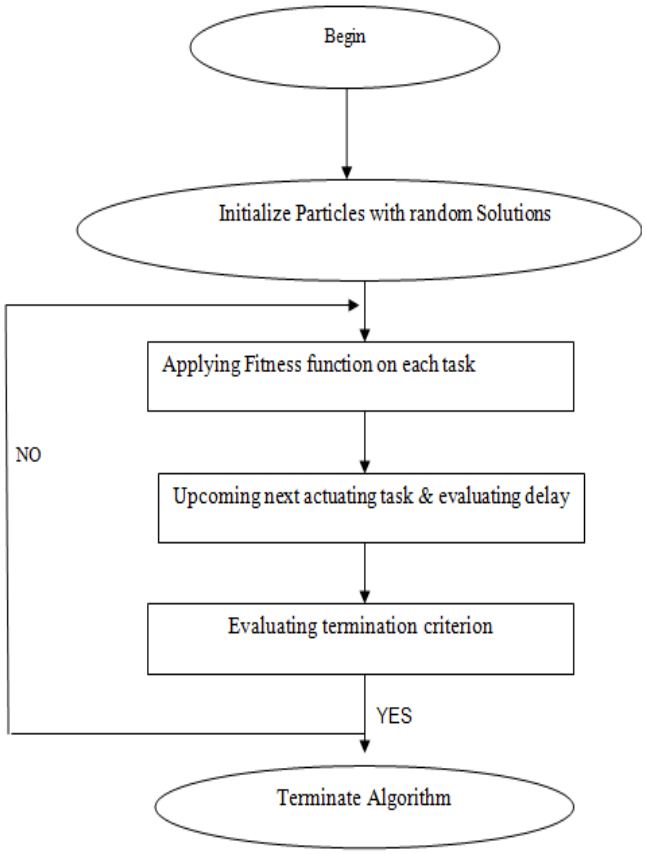

Fig.2 Flowchart of MFA algorithm

IV. PROPOSED MEASURES

Proposed extended MFA (PEMFA) Algorithm

1. Smallest Job to fastest processor: - In this algorithm, following steps are performed: At first, we sort existing jobs and existing processors in ascending order, sorting of processors is done on the basis of their processing power. After sorting jobs and processors, jobs are assigned based on one-to-one mapping.

2. Merging SJFP in MFA algorithm: - In standard MFA algorithm initial jobs are created randomly, but randomness decreases the chance of algorithm to converge to best solution, in order to improve the behavior of MFA algorithm, we merge shortest job to fastest processor algorithm(SJFP) into MFA, i.e. instead of generating initial population randomly we create them considering SJFP algorithm. All other steps are similar to standard MFA algorithm, so the flowchart of this modified MFA algorithm is as Fig. 3.

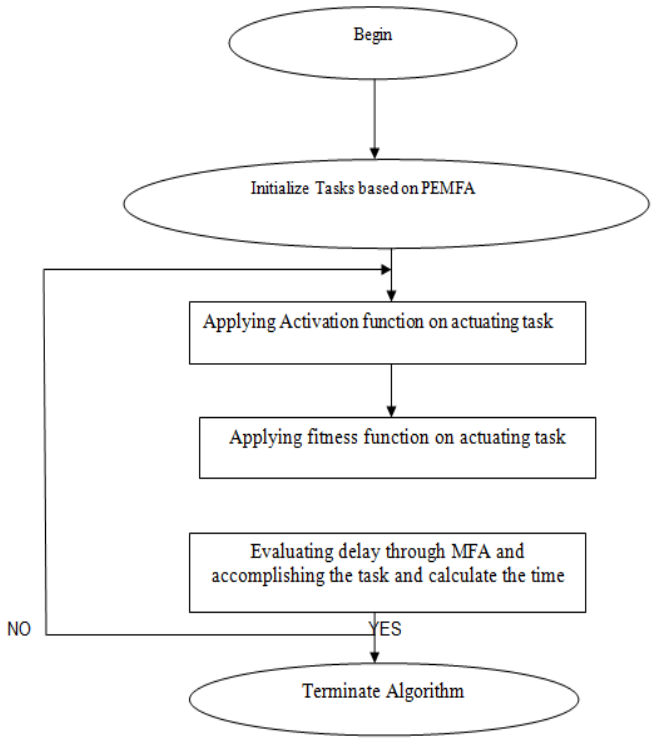

Fig. 3: Flowchart of the proposed EXTENDED Modified Filtering Algorithm

V. MODIFIED FILTERING ALGORITHM

The algorithm of PEMFA

Begin

Step 1: Initialization. Set the generation counter ω with i = 1 and j = 1;

Initialize the task of i to n and j to p randomly to each task corresponding to a potential solution to the given problem;

define light absorption coefficient ω;

set controlling the step size X_i and the initial attractiveness ω_o.

Step 2: Evaluate the time execution for each task with respect to machine in i and j iterations

Step 3: while the termination criteria is not satisfied or G < MaxGeneration do

for i=1:n (all n tasks)

do

for j=1:n (n tasks)

do

if (Xj <Xj), move task i towards j;

end if

there is delay in execution

Evaluate new execution time and update time of accomplishment;

end for j

end for i

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

$\omega_0 = \omega_{0+1}+1;$

Step 4: end while

Step 5: post-processing the results and visualization;

End.

Algorithm 1. The Algorithm of Modified Filtering Algorithm

VI. EXPERIMENTS AND RESULTS

We have used 4 VMs for evaluating performance of these algorithms, each VM has 4 GB of RAM and 30 GB of hard disk and 1 up to 4 cores of CPUs, in which each core has 2.3 GHz processing speed. In order to examine the performance of the proposed algorithm we have implemented proposed MFA algorithm. In this task, scheduling problem the criterion that is considered for evaluating performance is Makespan. To test the performance of this algorithm we have tested two different scenarios. In the first scenario, we changed number of tasks from 100 to 800. The second scenario is performed by changing number of iterations for constant number of tasks. Each experiment is repeated 10 times, and final result is the average of all 10 iterations. The criterion that is considered for the performance of this algorithm is Makespan. The improvement percentage is calculated by (7) [11]:

$$\phi = (1 - \frac{\sum_{J}^{I} \omega Makespan_{PSO}}{\sum_{j-1}^{i-1} \omega{-}1 Makespan_{MFA}}) * 100 (7)$$

Table I shows the parameters that are used for proposed MFA, and Table II shows the parameters that are used for PSO and modified PSO in the first scenario:

Makespan Algorithm: -

Input.	J, M, pi,j for all i ∈ M and j ∈ J
Output	xi,j ∈ {0, 1} for all i ∈ M and j ∈ J
Step 1	By binary search in [α/m, α] compute smallest value T of the parameter t such that lp(t) has a feasible solution
Step 2	Let x be an extreme point solution for lp(T).
Step 3	Construct graph H and find perfect matching P
Step 4	Round in x all fractionally set jobs according to the matching P

In this algorithm, we consider the classical Makespan Scheduling problem.

We are given m machines for scheduling, indexed by the set M = {1, . . . , m}.

There are furthermore given n jobs, indexed by the set J = {1, . . . , n}, where job j takes pi,j units of time if scheduled on machine i. Let Ji be the set of jobs scheduled on machine i. Then

$$\ell i = \sum_{j \in Ji} P$$

where

pi,j is the load of machine i.

The maximum load ℓmax = cmax = maxi∈M ℓi is called the makespan of the schedule

Results for three algorithms are shown in Table III.

Table 1: The modified MFA parameters

Population Size	60
Cross over Probability	**.81**
Mutation Probability	.01
Maximum no of Iteration	10

Table II: PSO and Modified PSO Parameters

Population Size	60
W	**0.65**
C1	2
C2	2
Maximum no of Iteration	10

Concluding results for three algorithms are shown in Table III

Scheduling Algorithm	Number of tasks	Number of resource	Make Span	Improvement
PSO	100	4	659	3.36
MODIFIED PSO			665	
MFA			679	
PEFMA			588	
PSO	200	4	1080	12.32
MODIFIED PSO			1194	
MFA			1199	
PSO	400	4	2002	3.02
PEFMA			982	
MODIFIED PSO			1988	
MFA			2036	
PSO	600	4	3315	3.72
MODIFIED PSO			3225	
MFA			3389	
PEFMA			2987	
PSO	800	4	4712	6.83
MODIFIED PSO			4679	
MFA			4898	
PEFMA			4056	

PEMFA

In this paper, PEMFA has been referred as the proposed filter, which yields out the results as the best among all most popular filters used before. PEMFA in this paper is fundamentally used with least possible assumptions; rhe main effectiveness of PEMFA is result in degradation of MALESPAN. The results from the concluding table shows that PEMFA filtering comes out to be one of the important filtering algorithm which remove assumptions based concept and make the MAKESPAN much lower. But some issues related to previous algorithm is that they have degraded the performance of parameters that they have use (Please refer to the Introduction and Related works).

To avoid these issues, there are two modifications in this algorithm which improve delay and as well as effective completion of time of tasks assigned in cloud and hence

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

improve the performance of the PEMFA algorithm.

The performance of these three algorithms based on noted values is shown in Fig. 4, in which x-axis represents number of tasks and y-axis shows Makespan.

VII. COMPARATIVE ANALYSIS

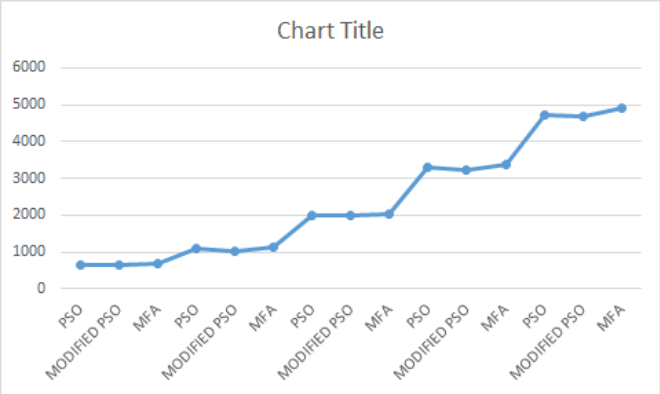

Fig. 4: Comparing performance of different algorithms based on noted values

As it is shown in the Table III and Fig. 4, we have performed experiments for 100,200,400 and 800 tasks on 4 VMs. The criterion that is considered for evaluating performance of these algorithms is Makespan i.e. the time in which execution of last task on processing machines becomes completed. Based on presented results in table III and Fig. 4 in comparison between PSO, modified PSO and MFA algorithm, in three sets of experiments MFA algorithm shows better result than PSO and modified PSO, but proposed modified MFA algorithm outperforms these two algorithms. In all sets of experiments, i.e. Makespan for this algorithm is has the smallest value in all experiments.

In the second scenario, all parameters are the same as table II and III except number of iterations, which is changed from 20 to 360 iterations. In this experiment number of tasks is constant and is equal to 400. Experimental results are shown in Fig. 5.

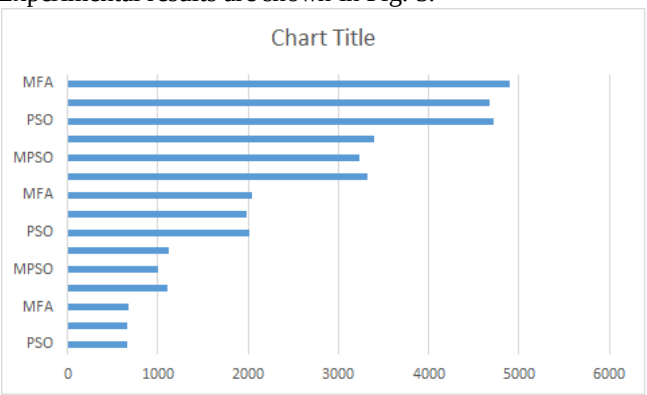

Fig. 5: Experimental results of different algorithms

As it is shown in Fig. 5 by incrementing number of generation Makespan decreases, but because of optimized initial jobs in the beginning of algorithm, modified MFA algorithm is able to produce solutions with more optimal Makespan.

VIII. CONCLUSIONS

In this paper, the problem of task scheduling in cloud computing environment is evaluated. Using modified MFA algorithm in comparison with PSO and modified PSO algorithms; which are still considered as the most efficient algorithms for scheduling tasks in distributed systems. In order to improve the performance of standard MFA algorithm the modified MFA algorithm is suggested, in which SJFP algorithm is merged into standard MFA algorithm for generating initial population in order to minimize Makespan. Our experiments depicts that even if both PSO and modified PSO algorithms and MFA algorithm show acceptable results, it can be said that by and large the proposed MFA algorithm shows better results than PSO and modified PSO algorithms. In short, the modified MFA 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.

REFERENCES

1. China 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 (2016). 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 2017.
4. Z. Yingfeng, L. Yulin, “Grid Computing Resource Management Scheduler Based on Evolution Algorithm [j]”, Computer Engineering Conference, 2016, 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 2015.
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, 2014.
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, 2017.
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, “Jobs swarm optimization”, Proc, IEEE Conf. Neural Netw., vol. IV, IEEE, Piscataway, NJ, 2014,pp.1942-1948.
10. L. Zhang, Y. Chen, B. Yang “Task Scheduling Based on MFA Algorithm in Computational Grid”, 2006 Proceedings of the 6th International Conference on Intelligent Systems Design and Applications, vol-2, 16-18 Oct, 2016,Jinan, China.
11. T. Chen, B. Zhang, X. Hao, Y. Dai, “Task scheduling in grid based on jobs swarm optimization”, The Fifth International Symposium on Parallel and Distributed Computing, ISPDC '06. pp. 238-245, 2016.

12. Widrow Bernard, Samuel Stearns D. Adaptive Signal Processing. – Pearson Education, Delhi, 2004.
13. Monson Hayes H. Statistical Digital Signal Processing and Modelling. – John Wiley & Sons Inc, Kundli, 2002.
14. Emmanuel Ifeachor C., Jervis Barrie W. Digital Signal Processing – A practical approach. – Pearson Education, Delhi, 2004.
15. Georgi Iliev, Nikola Kasabov. Adaptive filtering with averaging in noise cancellation for voice and speech recognition. – Department of Information Science, University of Otago, 2001.
16. Diniz P. Adaptive Filtering Algorithms and Implementation Issues. – Kluwer Academic Publishers, USA, 2002.
17. Xiao Hu, Ai-qun Hu, Li Zhao. A Robust Adaptive Speech Enhancement System // IEEE Int. Conference on Neural Networks and Signal processing. – Nanjing, China, December14-17, 2003.
18. Ikeda S., Sugiyama A. An adaptive noise canceller with low signal distortion for speech codecs // IEEE Trans. Signal Processing. – Vol. 47, Mar. 1999. – P. 665–674.
19. Julie E. Greenberg. Modified LMS Algorithms for Speech Processing with an Adaptive Noise Canceller // IEEE Trans. On Speech and Audio Processing. – Vol. 6, No. 4, July 1998.