Towards Performance Improvement of Cloud Applications under Virtualized Environment using PSO based K-means++ Algorithm

A.P.Nirmala[#1], Dr.R.Sridaran[*2],
[#1] Research Scholar, Karpagam University, Coimbatore & Assistant Professor,
New Horizon College of Engg.Bangalore, India.,
[#1] nirmalasuresh.ap@gmail.com
[*2] Dean, Faculty of Computer Applications,
Marwadi Education Foundation's Group of Institutions, Rajkot, India
[*2] sridaran.rajagopal@gmail.com

Abstract— Virtualization generally provides performance isolation between virtual machines running simultaneously on a single physical machine. But isolation does not happen to its fullest extent which may lead to performance interference. Due to this, the performance of applications running on one VM may get affected by running applications of co-existing VMs. This further leads to delay in execution time. As a result of this, throughput of cloud applications will be degraded. In order to deal with this situation, the proposed work has been presented which uses Particle Swarm Optimization based k-means++ algorithm for its implementation. The proposed work has been compared with some of the existing approaches and found to have better throughput and reduction in run time and cost.

Keyword- PSO based k-means++, Performance interference, Throughput, Virtualization, Scheduling, Fast genetic k-means++

I. INTRODUCTION

Virtualization is one of the most significant topics in the field of information technology. Server virtualization is the commonly used forms of the virtualization in Cloud computing (CC) [16]. In server virtualization, a single physical system is partitioned into multiple, independent logical systems. Thus, it provides flexibility for a single physical server to be used as multiple Virtual Machines (VMs).

VM software, a hypervisor or a virtual machine monitor (VMM) that runs on a server under virtualized environment. It enables a single physical machine to support a number of guest operating systems (OSs), each of which runs on its own virtual instance of the underlying physical machine. For example, Linux, Windows, Mac OS and Solaris can run on separate VMs within a single physical virtual server platform [19][5] and this is depicted in Fig.1.

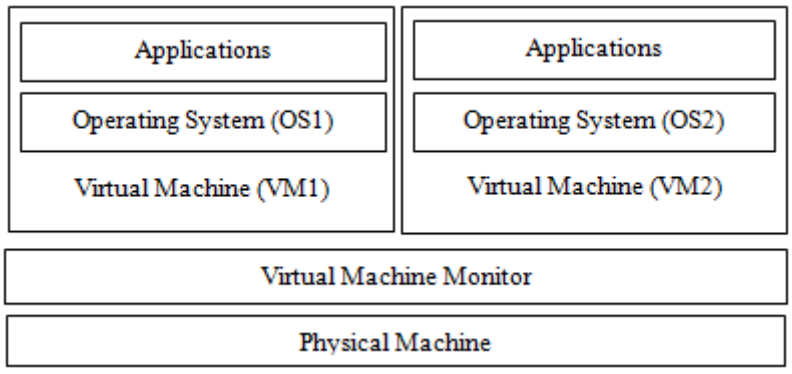

Fig. 1 Virtualized cloud environment

Virtualization has many advantages including the ability to slice a physical machine in terms of CPU and memory space. Despite of its advantages, there is a need for fault and performance isolation against the unexpected behaviors among the VMM and VMs. In fault isolation, faults and misbehaviors in one VM are not transmitted either to the hypervisor or other VMs hence it is well addressed by existing virtualization techniques.

In performance isolation, a VMM manages the VMs running on a single platform and also guarantees that they are functionally isolated from each other. However, Qun Huang et. al [2] enlightened that it could be

difficult to achieve in full extent. Hence, virtualization does not provide effective performance isolation between VMs. The same is emphasized by Koh et. al [3] that the performance of one virtual machine might vary at different times depends on other VMs in the physical host. Thus it leads to performance interference between the VMs that are co-located at each physical machine (PM). As all VMs share the same physical resources, they also mutually influence each other. According to X. Pu et. al [5], VMMs (hypervisors) have the ability to slice down resources and allocate the shares to different VMs where the applications running on one VM may still affect the performance of applications running on its neighbourhood VMs. As a result, interference might be a barrier in attracting performance sensitive customers. Further, interference exists when the co-located VMs are concurrently competing for hardware resources. Cloud users may suffer from performance degradation in terms of slowdown of tasks' execution time. This may reduce the rate of completion jobs within their deadlines.

Hence, this becomes the need of the hour to develop architectural techniques that would ensure proper sharing of resources allocated to VMs running simultaneously in the physical machine. Effective management of virtualized cloud environments introduces new and unique challenges such as efficient CPU scheduling for VMs, effective allocation of VMs to handle both CPU intensive and I/O intensive workloads. Based on this, many novel scheduling algorithms can be thought of. The primary objectives of these algorithms is to either minimizing the negative impacts of co-located applications or improving the overall system performance based on throughput [4], [1]. Ron C. Chiang et. al [1] discusses how the system can make optimized scheduling decisions that would lead to significant improvements in both application performance and resource utilization.

K-means algorithm is an iterative method of cluster analysis. In the iterative process, the distance between the observation point and cluster center is to be minimized. The limitation with this algorithm is, it may converge to local minimum if no proper initialization of cluster center is made. Since K-means algorithm converges to a local minimum, Genetic Algorithm (GA) is used for the purpose of finding the global minima [13], [14]. According to Ron C. Chiang et. al in [1], since k-means++ algorithm makes a good choice of initial *k* centers it can be utilized to replace k-means algorithm. K-means++ picks each point at random with probability proportional to the squared distance. Thus k-means++ is combined with genetic algorithm to find optimized solution. This motivates us to propose scheduling algorithms to improve the performance in terms of throughput, runtime and also cost in the virtualized environments. Hence, fast genetic k-means++ algorithm was proposed and it was implemented by conducting a comprehensive evaluation with a variety of cloud applications in our previous work [26].

As Particle Swarm Optimization (PSO) is a popular algorithm [8], [9], [10], [11] and it has a faster convergence rate than GA, PSO algorithm is proposed for finding the global optimal solution. Extension to our previous work, the proposed PSO based k-means++ algorithm aims at improving the performance by scheduling the task to various VMs with minimized interference occurs from co-located applications. The algorithm is designed in a way that it reduces the runtime and improves the I/O throughput for data-intensive applications in a virtualized environment. As soon as the task arrives, the scheduler generates a number of possible assignments based on the incoming tasks and list of available VMs. Then the scheduler makes the decision for scheduling and assigning the task to different servers based on the predictions.

This paper is organized as follows: section II provides the related work, section III explains the proposed methodology and experimental results are discussed in section IV. Section V gives the conclusion with future work.

II. RELATED WORK

X. Pu et al. [5] have discussed the performance interference among VMs which run the network workloads in virtualized environments. In their work, the authors have considered certain system-level characteristics as metrics to identify the impact on the throughput when running different combinations of workloads with different file sizes. Many experiments are conducted for each combination of workloads and factors that may lead to performance interferences are found out. But the above work does not describe how to mitigate the I/O interference for data-intensive applications. Several research works are available to assess the performance degradation of VMs due to interference which are illustrated in [7], [3]. Though these studies propose the different types of benchmarks relating to VM interference, mitigation of interference effects are not analysed.

DeepDive is a system for VM migration proposed by D. Novakovic et. al [19]. It recognizes interference-inducing VM and migrates that to destination physical machine where least interference is observed. In DeepDive, the authors mainly considered the placement of VMs in order to reduce the interferences.

In Paragon [20], test benchmarks are proposed to detect the interferences and their impacts on co-located applications. It uses past-scheduled applications to decide on the best placement for any new application with respect to interference in the place of profiling. Instead of scheduling the applications in a VM, Paragon classifies and schedules the application on a hardware platform itself to mitigate interference.

Altino Sampaio et. al in [21] have discussed the algorithms which are used to schedule VMs dynamically in order to mitigate the performance interference due to hardware resources such as last-level cache(LLC) sharing.

The authors intend to maximize the rate of completed tasks by constructing performance-efficient computing environments that reacts to performance degradation arise from sharing of LLC memory. QoS-aware control framework, Q-clouds [18] is developed to minimize performance interference effects. It uses a multi-input multi-output (MIMO) model that tunes resource allocations to capture the performance interference in a virtualized environment and focuses only on the CPU bound workload.

According to [29], [30], PSO is considered as more suitable algorithm in cloud computing. Since the initial particles are created randomly in PSO algorithm, this leads to reduce the opportunity of converging to best solution. Solmaz Abdi et. al [22] proposed, PSO algorithm merged with shortest job to fastest processor algorithm (SJFP) in order to improve the performance of the PSO algorithm.. Shaobin Zhan et. al in [24] involved simulated annealing with PSO in each iteration, in order to achieve fast convergence rate. Thus, PSO algorithm can be improved in performance by merging it with suitable algorithms [22], [24], [27], [28].

Saurabh Bilgaiyan et al. [30], focused on the need of proper scheduling in cloud computing environment, as the process of scheduling is important to manage resources, minimize the idle time and increase the performance of systems. In their work, the authors have analyzed the different evolutionary and swarm based algorithms for task scheduling by discussing their advantages and the mechanism used in those algorithms.

Shengjun Xue et. al [23], have discussed that improved ant colony optimization scheduling algorithm can allocate the resources for tasks and improve the utilization rate of resources. The authors in their work focused on reducing the execution time and also avoiding the resource wastage by balancing the number of tasks assigned to virtual machines based on its capability. But they do not focus on the cost issues. Gang Zhao [25] highlights the importance of considering the usage cost of resources. He has further proposed fitness function to achieve a balance between the minimization of processing time and total cost but throughput is not taken into account.

K. Krishna et. al [13] have illustrated that, K-means algorithm is the simplest and the most popular clustering algorithm but it may converge to local optimal solution. Hence the authors have proposed that genetic k-means algorithm finds the global optima. Ron C. Chiang et. al [1] further proves that k-means++ algorithm picks each point at random with probability proportional to the squared distance instead of choosing the point farthest from the already chosen points. Thus k-means++ is combined with genetic algorithm to find optimized solution in our previous work [26].

Our proposed work extends this phenomenon using PSO based k-means++ algorithm in place of genetic algorithm for finding global optimal solution. It is similar to Paragon [20], as far as the placement of application is concerned in scheduling. In contrast to [18], [19], our proposed work focuses on data-intensive applications. It is essential to address the challenges of the I/O interference when running data-intensive applications as cloud applications are data centric in a virtualized environment. Contrary to DeepDive [19], it considers the placement of tasks in appropriate VMs to reduce the interferences. PSO based k-means++ is a combination of PSO algorithm with k-means++ is proposed to make the optimized decisions in order to improve the overall performance by considering not only the time and cost but also the throughput, unlike the authors in [23], [25], as they considered cost and time only.

III. PROPOSED WORK

The proposed work uses PSO based k-means++ scheduling algorithm to improve the application performance in a virtualized environment. Further, Levenberg-Marquardt method [17] has been applied as the Interference Prediction Model (IPM) which works by collecting the application performance from the resource consumption observed from multiple VMs. This model is built with five parameters (controllers) denoting CPU utilization in VMM, CPU consumption from data processing of application, I/O request, cost and job per cloudlet. Interference profile is generated by running the given application on one VM, while the remaining (n-1) VMs execute various workloads in the background where the n VMs being considered in the virtualized environment. This approach facilitates online learning of IPM that is dynamically modified and monitored for various applications in the cloud platform. Further, Interference Aware Scheduling (IAS) aims at reducing the runtime and improving the I/O throughput for data-intensive applications in a virtualized environment. It schedules and allocates the tasks to various VMs with minimized interference occurring from co-located applications.

PSO based k-means++ algorithm has been applied for the purpose of improving the performance of the applications in the cloud environment. When the task arrives, the scheduler continues to generate different possible assignments of the same, based on the incoming tasks and list of available VMs. Afterwards, the scheduling process is carried out by assigning the tasks to different servers based on the predictions. The performance of the above algorithm has been tested on the application throughput, runtime and cost whose results observed are widely discussed under section IV.

A. PSO based k-means++

K-Means algorithm (KMA) is an iterative algorithm which starts with initialization of cluster centers [11]. This algorithm aims at partitioning n observations (objects) into k clusters. Every observation point is assigned

to one cluster that has minimum distance between the observation point and the cluster center. This process is repeated in each iterations and the algorithm stops when the mean distance between the observation point and the cluster center is minimized or when fixed number iterations are reached.

The major weakness with KMA is its tendency to converge to a local minimum in case of no proper selection of initial partition is made and this may have an influence on the final solution. This is avoided through certain optimization approaches like GA and PSO. One more weakness of KMA is that, with respect to its choice of points that are far away from each other. On the other hand, K-means++ algorithm tends to make a good choice of initial *k* centers to overcome the drawback of KMA. Moreover, it will pick each point at random with the probability proportional to the squared distance rather than the point farthest [12]. Due to these two reasons, K-means++ has been used in the proposed work.

Fast k-means++ algorithm was proposed in our previous work [26] to overcome these drawbacks by combining GA with k-means++. Similar to GA, PSO is another popular algorithm which has the capability to achieve global optimistic solution. But PSO has a faster convergence rate than GA. In addition, PSO has fewer mathematical operators compared to GA which makes the applications less dependent on modifying those operators. PSO is faster than GA in assigning the tasks and also gives better schedule than GA in distributed computing systems and Grid computing. In K-means++, convergence rate is faster in finding local optimum solution whereas it is weaker in finding the global solution and thus motivates the combination of PSO and k-means++ by considering the advantages of both the algorithms in virtualized environment [10].

PSO based k-means++ algorithm is proposed by combining the ability of the globalized searching of PSO and the fast convergence of k-means++ as our proposed work revolves around the cloud computing environment. PSO based k-means++ algorithm consists of two modules namely PSO and K-means++. At the initial stage, the PSO module is executed for a short period to search for the clusters' centroid locations. The locations are given as input to the K-means++ module for refining and generating the final optimal clustering solution.

1) *PSO module*: PSO is an optimization algorithm which simulates the behavioral patterns of bird flock. In a flock of birds moving over an area in search of food, the one which is nearest to the food makes a sound and the other birds start to move forward and backward in its direction. In this process, if any other bird in the flock comes closer to the food than the first, it signals the others in the flock and those birds start to move towards that. This process continues till one of the birds reaches the food [6]. A similar approach has been adapted in the proposed work for assigning the tasks to suitable VMs.

PSO module consists of particles (tasks) that constitute a swarm, moving around the search space looking for the best solution. Each particle keeps track of its coordinates in the solution space which are associated with the best solution (fitness) that has been achieved so far by that particle. This value is called personal best, **pbest**. If any neighborhood of the particle obtains best value than the pbest, then this value is called global best, gbest. The basic concept of PSO lies in accelerating each particle towards its pbest and the gbest locations, with a random weighted acceleration at each time [6]. Each particle keeps changing its position based on the distance between the i) current position and **pbest** and ii) current position and **gbest**

Each particle can be shown by its current speed and position, the most optimist position of each individual and its surrounding. The speed and position of ith particle changes according to the following equations

$$v_{id} = w^{*} v_{id} + c_1 {}^{*} rand_1 {}^{*} (p_{id} - x_{id}) + c_2 {}^{*} rand_2 {}^{*} (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

where w denotes the inertia weight factor; P_{id} the location of the particle that experiences the best fitness value; P_{gd} the location of the particles that experience a global best fitness value and $c_1$and $c_2$are constants that are known as acceleration coefficients, d the dimension of the problem space, $rand_1$and $rand_2$ are random values in the range of (0, 1). The inertia weight factor w provides the necessary diversity to the swarm by changing the momentum of particles to avoid the stagnation of particles at the local optima.

Equation 1 requires each particle to record its current coordinateX_{id}, its velocity V_{id} that indicates the speed of its movement along the dimensions in a problem space and the coordinates $P_{id}$and $P_{gd}$where the best fitness values were computed. The best fitness values are updated at each generation, based on the following equation:

$$P_i(t+1) = \begin{cases} P_i(t) & f(X_i(t+1)) \le f(X_i(t)) \\ X_i(t+1) & f(X_i(t+1)) > f(X_i(t)) \end{cases} \quad (3)$$

In the above equation, variable f indicates fitness function and P_i(t) indicates best fitness values and the coordinates, where the value was calculated and t denotes the generation step.

Each particle maintains a matrix X_i= (C_1, C_2, …, C_i, .., C_k), where C_i represents the i^{th}cluster centroid vector and k is the cluster number. For each iteration, the particle adjusts the position of the centroid vector in the vector space according to its own experience and those of its neighbors. The average distance between a

centroid of the cluster and a point is used as the fitness value to evaluate the solution represented by each particle. This fitness value is measured by the equation given below:

$$f = \frac{\sum_{i=1}^{N_c} \{ \frac{\sum_{j=1}^{p_i} d(o_i, \mathrm{m}_{ij})}{p_i} \}}{N_c} \quad (4)$$

where m_{ij} denotes the j[th] point vector that belongs to cluster i, O_i the centroid vector of i[th] cluster, d(O_i, m_{ij}) the distance between point m_{ij} and the cluster centroid O_i, $P_i$stands for the point number which belongs to cluster O_i and N_c stands for the cluster number [6].

2) *K-means++ module:* The K-means++ module will inherit the result of the PSO module as the centroids for initial clustering and will continue processing the optimal ones by recalculating them based on equation 5, to generate the final result.

$$c_j = \frac{1}{n_j} \sum_{\forall d_j \in S_j} d_j \quad (5)$$

where d_j denotes the point vector that belongs to cluster S_j, c_j for the centroid vector and n_j is the number of point vectors that belong to cluster S_j.

The algorithm given in Fig.2 provides the flow of PSO module and k-means++ module to find the optimized result.

Begin

1. t = 0
2. Initialize particle randomly
3. Evaluate fitness function P(t).
4. Finding pbest value
5. If the termination criterion is not satisfied, go to Step 3
6. Update the velocity and position of the particle
7. Finding gbest value, the optimized solution by K-means++ as given below.

    a) The data set is reassigned to clusters according to nearest rule.
    b) Cluster centroids, fitness value are recalculated and positions are updated
8. If the position is satisfactory or the maximum iteration is reached, then process is stopped. Otherwise t = t + 1, go to Step 3.

End

Fig. 2 Pseudo code of PSO based k-means++

IV. EXPERIMENTAL RESULTS & DISCUSSION

The proposed algorithm is evaluated and compared with i) Fast genetic K-means++ and ii) K-means++ algorithm [1], [26] by measuring the performances based on the cost, throughput and run time. In order to generate a more realistic workload, we randomly choose the data sets, data sizes and number of processes. From the comparison of results, the improvements in the performance of virtualized environment are illustrated.

A. PSO based k-means++

The experimental setup that has been used for the implementation consists of Intel Core2 duo CPU with 3.40 GHz speed, 4GB RAM and 500 GB capacity hard disk. The software tools used are Java for coding the simulator, Mysql for data store and Cloudsim [15] for simulating the virtualized environment. For the purpose of testing the algorithms, files various types such as pdf, image and text are used.

B. Performance evaluation

The application throughput is evaluated in terms of number of tasks completed per unit time. When the task arrives, the scheduler decides the scheduling and assigning the tasks to VM based on minimum CPU utilization using different scheduling algorithms such as k-means++, fast genetic k-means++ and PSO based k-means++ algorithm. The normalized application throughput is measured for all these algorithms. The observed throughput, running cost and the run execution time taken up by the cloud application such as pdf, text and image files in the

experiment are shown in Fig.3&4. It is found that, PSO based k-means++ achieves better throughput than the other two namely fast genetic k-means++ and k-means++ algorithm as in Fig 3.

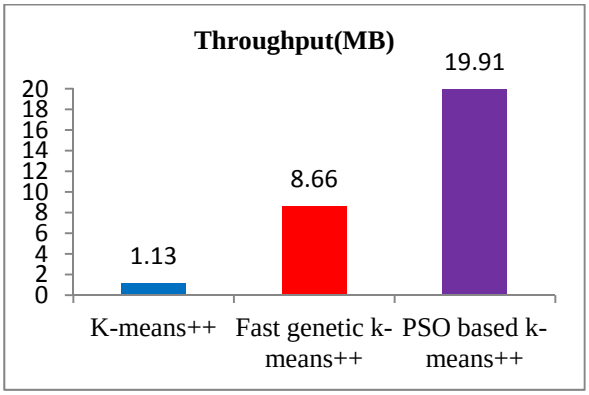

Fig. 3: Throughput Comparison

The cost for running the cloud application is calculated based on cloud usage time. The cost is handled by performing the function with the minimal running time. Based on the consumption, the cost is computed as Rs.1 per second. So, the cost for k-means++ is $42 \times 1 = \text{Rs.}\,42$, the cost of Fast genetic k-means++ is $23 \times 1 = \text{Rs.}\,23$, whereas the cost for the PSO based k-means++ is $10 \times 1 = \text{Rs.}\,10$. Hence the proposed technique takes the least time and lowest cost while compared with existing techniques. The comparison of three algorithms for the same cloud data is shown in Fig.4.

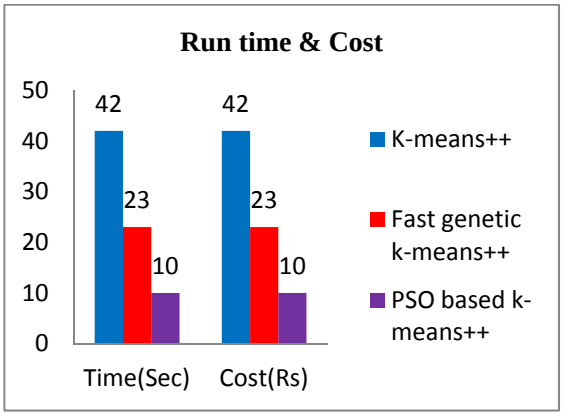

Fig. 4: Comparison of Time and Cost

V. Conclusion

The proposed work has been proven to gain the combined advantages of particle swarm and k-means++ algorithms. The results show promising improvements in terms of increased throughput, reduce the runtime and cost over the existing approaches. This may be extended by considering certain other parameters like memory, disk usage and network I/O traffic.

References

[1] Ron C. Chiang and H. Howie Huang, "TRACON: Interference-Aware Scheduling for Data-Intensive Applications in Virtualized Environments", IEEE Transactions On Parallel And Distributed Systems, IEEE 2013.

[2] Qun Huang and Patrick P. C. Lee," An Experimental Study of Cascading Performance Interference in a Virtualized Environment", ACM SIGMETRICS Performance Evaluation Review, Volume 40 Issue 4, March 2013.

[3] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu. An Analysis of Performance Interference Effects in Virtual Environments. IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2007, pp. 200-209.

[4] Ron C. Chiang and H. Howie Huang. TRACON: Interference-aware scheduling for data-intensive applications in virtualized environments. In Proc. Of SC, pages 1 –12, nov. 2011.

[5] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu. Understanding performance interference of I/O workload in virtualized cloud environments. In Proc. of CLOUD, pages 51 –58, july 2010.

[6] J Kennedy, and RC Eberhart, "Particle Swarm Optimization," in Proc. the IEEE International Joint Conference on Neural Networks, Vol. 4, pp 1942–1948, 1995.

[7] H. Shan, K. Antypas, and J. Shalf. Characterizing and predicting the I/O performance of HPC applications using a parameterized synthetic benchmark. In Proc. of SC, pages 42:1–42:12, Piscataway, NJ, USA, 2008. IEEE Press.

[8] Nadjet Kamel, Imane Ouchen, Karim Baali, "A Sampling-PSO-K-means Algorithm for Document Clustering", Genetic and Evolutionary Computing Advances in Intelligent Systems and Computing Volume 238, 2014, pp 45-54.

[9] Alireza Ahmadyfard, Hamidreza Modares, "Combining PSO and k-means to Enhance Data Clustering ",2008 Internatioal Symposium on Telecommunications.

[10] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, Rajkumar Buyya. " A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments", Advanced information networking and applications (AINA), 2010 24th IEEE International conference.
[11] Mehdi Neshat, Shima Farshchian Yazdi, Daneyal Yazdani and Mehdi Sargolzaei, "A New Cooperative Algorithm Based on PSO and K-Means for Data Clustering", Journal of Computer Science 8 (2): 188-194, 2012 , ISSN 1549-3636, Science Publications.
[12] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In SODA'07.
[13] K. Krishna and M. NarasimhaMurty, "Genetic K- Means Algorithm", IEEE Transactions On Systems, Man And Cybernetics—Part B:Cybernetics, Vol. 29, No. 3, June 1999.
[14] Yi Lu, Shiyong Lu, FarshadFotouhi, Youping Deng, Susan J. Brown," FGKA: A Fast Genetic K- means Clustering Algorithm", ACM, 2004.
[15] RajkumarBuyya, Rajiv Ranjan, Rodrigo N. Calheiros,"Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges And Opportunities", in The 2009 International Conference onHigh PerformanceComputing and Simulation, HPCS2009, pp:1-11.
[16] Ravi Iyer, Ramesh Illikkal, OmeshTickoo, Li Zhao, Padma Apparao, Don Newell, "VM3: Measuring, modeling and managing VM shared resources", 2009 Elsevier Computer Networks 53 (2009) 2873–2887.
[17] K. Madsen, H.B. Nielsen, O. Tingleff, "Methods for Non-linear Least Squares Problems Informatics and Mathematical Modelling", Technical University of Denmark. April 2004.
[18] R. Nathuji, A. Kansal and A. Ghaffarkhah, " Q-Clouds : managing Performance interference effects for qos-aware clouds". In EuroSys '10.
[19] D. Novakovi´c, NedeljkoVasi´c, Stanko Novakovi´c, DejanKosti´c, and Ricardo Bianchini. DeepDive: Transparently Identifying and Managing Performance Interference in Virtualized Environments. Technical Report 183449, EPFL, 2013.
[20] C. Delimitrou et al. Paragon: QoS-aware scheduling for heterogeneous data centers. In ASPLOS, 2013.
[21] Altino Sampaio, Jorge G. Barbosa, Parallel & Cloud computing, PCC Vol. 2 Iss. 4, 2013 PP. 116-125 www.vkingpub.com © 2013 American V-King Scientific Publishing.
[22] Solmaz Abdi, Seyyed Ahmad Motamedi, and Saeed Sharifian, "Task Scheduling using Modified PSO Algorithm in Cloud Computing Environment", International Conference on Machine Learning, Electrical and Mechanical Engineering (ICMLEME'2014) Jan. 8-9, 2014 Dubai (UAE).
[23] Shengjun Xue, Mengying Li, Xiaolong Xu, and Jingyi Chen, "An ACO-LB Algorithm for Task Scheduling in the Cloud Environment", JOURNAL OF SOFTWARE, VOL. 9, NO. 2, FEBRUARY 2014.
[24] Shaobin Zhan, Hongying Huo, "Improved PSO-based Task Scheduling Algorithm in Cloud Computing", Journal of Information & Computational Science 9: 13 (2012) 3821–3829.
[25] Gang Zhao , Cost-Aware Scheduling Algorithm Based on PSO in Cloud Computing Environment , International Journal of Grid and Distributed Computing Vol.7, No.1 (2014), pp.33-42 ISSN: 2005-4262.
[26] A.P.Nirmala, Dr. R. Sridaran,"A Novel architecture for improving performance under virtualized environments" (IJCSIS) International Journal of Computer Science and Information Security, Vol. 13, No. 1, January 2015.
[27] Xiaohui Cui and Thomas E. Potok, "Document clustering analysis based on hybrid PSO+K-means algorithm", Journal of Computer Sciences (special issue), 7-33, 2005. ISSN 1549- 3636.
[28] Sheng-Jun Xue, Wu Wu, "Scheduling Workflow in Cloud Computing Based on Hybrid Particle Swarm Algorithm", TELKOMNIKA Indonesian Journal of Electrical Engineering, vol. 10, no. 7, (2012), pp. 1560-1566.
[29] L Guo, S Zhao, S Shen, C Jiang , "Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm", - Journal of Networks, 2012.
[30] Saurabh Bilgaiyan, Santwana Sagnika, Madhabananda Das, "An Analysis of Task Scheduling in Cloud Computing using Evolutionary and Swarm-based Algorithms", International Journal of Computer Applications (0975 – 8887) Volume 89 – No.2, March 2014.

AUTHOR PROFILE

A.P.Nirmala received her MCA and M. Phil., degree in Computer Science from Bharathiar University, Coimbatore in 2000 and 2006 respectively. She is pursuing her Ph.D at Karpagam University, Coimbatore, India. She is presently working as Assistant Professor in the Department of MCA, New Horizon College of Engineering, Bangalore, India. She has 13 years of teaching experience. Her research area is Cloud Computing. She has published 2 research papers in International Journals and presented 8 research papers in National Conferences.

Dr.R.Sridaran is a Ph.D in Computer Science from Madurai Kamaraj University. He has published 25+ research papers in leading journals and presented in many conferences. He is presently guiding eight research scholars in the areas of Cloud Computing, e-learning and Software Engineering. He has got 22 years of academic experience and served in leading educational institutions at different capacities. He is currently the Dean of Faculty of Computer Applications, Marwadi Education Foundation, Rajkot. He is also the Chairman, Computer Society of India Chapter of Rajkot.