Study and Impact of CloudSim on the run of PSO in Cloud Environment

Kavita Bhatt
Department of Computer Science and Engineering
Poornima College of Engg, Jaipur ,Rajasthan, India

Dr. Mahesh Bundele
Department of Computer Science and Engineering,
Poornima Group of Colleges Jaipur, Rajasthan, India

Abstract- There is a concept of PSO algorithm which is very much efficient and effective with optimized result popular these days in so many streams. This paper provides the run of PSO on cloudSim with comparison analysis from different simulators. The paper covers five different sections in details. First is introduction towards the topic. Then cloud computing with cloudSim comes as another section which will be carried out with PSO as next section for the paper. PSO is based on swarm intelligence and is a good algorithm to find the near optimum. The algorithm is good for searching so as for load balancing even workflow scheduling can also be perform via PSO with good and efficient results. The fourth section of paper gives the explanation of execution of PSO over cloudSim with elaboration of graph results and finally the conclusion and future scope of the proposed concept covers in the last section.

Keywords - PSO, Cloud Computing, Cloud Model, VM, CloudSim.

I. INTRODUCTION

The starting six decades of Information Technology (IT) have witnessed a great evolution across several dimensions like: Infrastructure paradigms as centralized computing, personal computing, client-server computing, the Internet and mobile computing. Technology abstractions like operating systems, database and transaction processing, application components and service oriented architectures. Application domains like personal productivity and collaboration, industry-specific platforms and enterprise applications. Service models like infrastructure services, application management services, systems integration and consulting services.
The main idea for the development of cloud was to decentralize the application from hardware and OS. This technology allows for much more efficient computing by centralizing storage, memory, processing and bandwidth. Cloud is a style of computing which is having dynamically scalable virtualized resources provided as a service over the Internet-the web of networks. It reduces the time required to procure heavy resources and boot new server instances in minutes, allowing one to quickly scale capacity, both up and down, as ones requirement changes.
PSO is a simple algorithm based on swarm intelligence which is used for solving many issues. It is mainly used to optimize the output using mathematical formulas. PSO used in various domains to solve their problems like load balancing, workflow scheduling, searching, enhancing performance and so on. This algo can also be implemented in cloud and for this some changes are going to make in the steps of algorithm without changing the core concept. The algorithm has make successful run using different languages and tools like java,.NET, MATLAB, C++ and so on. Now we want to check that if it is executable on cloudSim or not? CloudSim is a toolkit for execution of cloud based applications or programs by providing them simulated environment without paying anything for using the Virtual machine and bandwidth etc form the system.
This research paper provides the PSO execution over the cloudsim toolkit with results in tabular form as well as graph and also proposed the search of cloudlets using PSO. The paper has carried out in next three sections, covered by cloud computing with cloudSim simulator, PSO and PSO with cloud, PSO implements on CloudSim, finally conclusion and future scope of the paper respectively.

II. CLOUD COMPUTING WITH THE SIMULATOR- CloudSim

Definition

Cloud computing is a technology that uses the internet and central remote servers to maintain data and applications. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access. This technology allows for much more efficient computing by centralizing storage, memory, processing and bandwidth. The areas in which cloud computing saves administrative costs include: Basic customization, Real-time reporting, Security and sharing models, multiple languages and currencies.

Cloud computing can be defined as “a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers”. Some examples of emerging Cloud computing infrastructures are Microsoft Azure, Amazon EC2, Google App Engine, and Aneka.

A cloud service has three distinct characteristics that differentiate it from traditional hosting:-On demand Services, Elasticity in execution of services and flexibility in user’s requirements.

Layered Pattern for Cloud Architecture

Figure. 1 shows the layered pattern of service-oriented Cloud architecture. In cloud architecture Software as a Service, Platform as a Services and Infrastructure as a Services, play major role as actually these are in the core of cloud architecture. Conventionally each component of cloud will run in a different VM, which can be hosted in data centers owned by different Cloud providers. **SaaS** is where application services are delivered over the network on a subscription and on-demand basis. Cisco WebEx™, Salesforce, Microsoft, and Google are a few providers in this layer. **PaaS** consists of run-time environments and software development frameworks and components delivered over the network on a pay-as-you-go basis. PaaS offerings are typically presented as API to consumers. Examples of this are: Google Apps Engine, Amazon Web Services, force.com, and Cisco® WebEx Connect. **IaaS** is where compute, network, and storage are delivered over the network on a pay-as-you-go basis. Amazon pioneered this with AWS (Amazon Web Service), and now IBM and HP are entrants here also. Cisco is also going to use this approach.

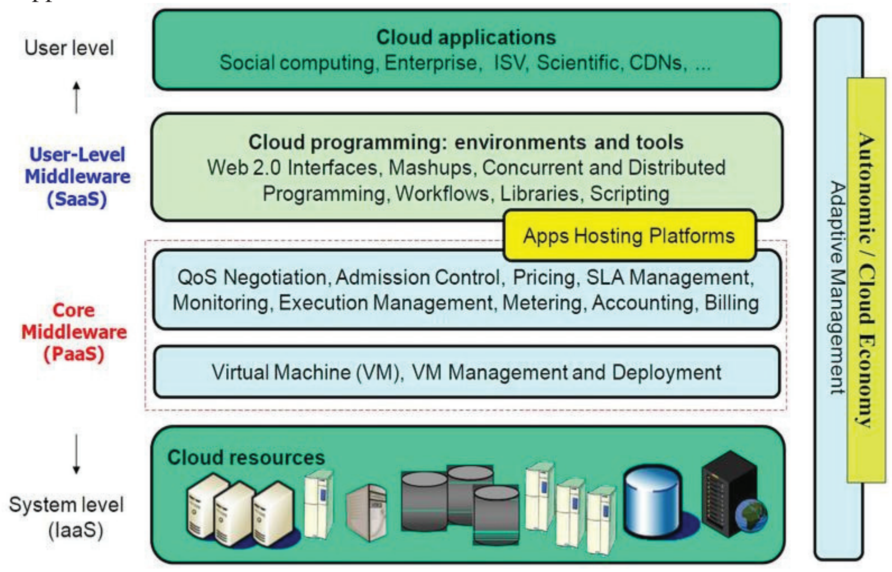

User level
User-Level Middleware (SaaS)
Core Middleware (PaaS)
System level (IaaS)
Cloud applications
Social computing, Enterprise, ISV, Scientific, CDNs, ...
Cloud programming: environments and tools
Web 2.0 Interfaces, Mashups, Concurrent and Distributed Programming, Workflows, Libraries, Scripting
Apps Hosting Platforms
QoS Negotiation, Admission Control, Pricing, SLA Management, Monitoring, Execution Management, Metering, Accounting, Billing
Virtual Machine (VM), VM Management and Deployment
Cloud resources
Autonomic / Cloud Economy
Adaptive Management

Figure. 1 Layered Pattern for Cloud Architecture

Category	Characteristics	Product Type	Vendors & Products
SaaS	Customers are provided with applications that are accessible anytime and from anywhere	Web applications and services (Web 2.0)	SalesForce.com (CRM), Clarizen.com (Project Management), Google Mail (Automation)
PaaS	Customers are provided with a platform for developing applications hosted on the Cloud	Programming APIs and frameworks; Deployment system.	Google AppEngine, Microsoft Azure, Manjrasoft Aneka
IaaS/HaaS	Customers are provided with virtualized hardware and storage on top of which they can build their infrastructure	Virtual machines management infrastructure, storage management	Amazon EC2 and S3; GoGrid; Nirvanix

Table 1. Categorization of Cloud Services

Why CloudSim?

The use of real infrastructures for the execution of our experiment such as Amazon EC2, limits the experiments to the scale of the infrastructure, and makes the reduplication of results very difficult for understanding. The main reason for this being the conditions prevailing in the Internet-based environments are beyond the control of developers of resource allocation and application scheduling algorithms. That's why we select such a toolkit which can survive in such conditions and hence we preferred CloudSim. This toolkit assisted the modeling and generation of virtual machines in a simulated manner such that datacenters, jobs and their mapping to VMs can be done on a same node whereas provide the desirable result. This toolkit can also be helpful when there are more than two datacenters exists or mapping of VMs are required.

The Cloudbus Toolkit is a collection of technologies and components that comprehensively try to address the challenges involved in making this vision a concrete reality. CloudSim is a component of Cloudbus but has important features. Among the unique features of CloudSim, there are: (i) availability of virtualization engine, which aids in creation and management of multiple, independent, and co-hosted virtualized services on a datacenter node; and (ii) flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

Modeling the Cloud: The core hardware infrastructure services are modeled by a Datacenter component for handling service requests. These requests are need to be allocated a share of processing power on Datacenter's host components. VM processing plays a novel role and by VM processing, we mean a set of operations related to VM life cycle: provisioning of a host to a VM, VM creation, VM destruction, and VM migration. Allocation of application-specific VMs to Hosts in a Cloud-based data center is the responsibility of the Virtual Machine Provisioner component. Each Host component instantiates a VM scheduler component which is responsible for implementing the space-shared or time-shared policies and for allocating cores to VMs.

Modeling the VM allocation: To allow simulation of different policies, CloudSim supports VM scheduling at two levels: First, at the host level and second, at the VM level. At the first level, it is possible to specify how much of the overall processing strength of each core in a host will be assigned to each VM in the simulation. At the second level, the VMs assign specific amount of the available processing strength to the idiomatic task units that are hosted within its execution engine.

Design and Implementation of CloudSim : CloudSim has several classes for solving number of purposes. Every DataCenter component instantiates a generalized resource provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices. DatacenterBroker class models a broker, which is responsible for mediating between users and service providers depending on users' QoS requirements and deploys service tasks across Clouds. SANStorage implements a simple interface that can be used to simulate storage and retrieval of any amount of data, at any time subject to the availability of network bandwidth. VirtualMachine class models an instance of a VM, whose management during its life cycle is the responsibility of the Host component.Cloudlet. This class models the Cloud-based application services (content delivery, social networking, business workflow), are commonly deployed in the DataCenterBroker .CloudCoordinator class is responsible for not only communicating with other peer CloudCoordinator services and DataCenterBroker, but also for monitoring the internal state of a data center that plays integral role in load-balancing/application scaling decision making. The function of BWProvisioner is to undertake the allocation of network bandwidths to set of competing VMs deployed across the data center. The execution and deployment of VM on a host is feasible only if the MemoryProvisioner component determines that the host has the amount of free memory, which is requested for the new VM deployment. The chief functionality of the VMProvisioner is to select available host in a data center, which meets the memory, storage, and availability requirement for a VM deployment. VMMAllocationPolicy is an abstract class implemented by a Host component that models the policies (space-shared, time-shared) required for allocating processing power to VMs.

Communication among Entities: Figure. 2 depicts the flow of communication among core CloudSim entities. In the beginning of the simulation, each Datacenter entity registers itself with the CIS (Cloud Information Service) Registry. Brokers acting on behalf of users consult the CIS service about the list of Clouds who offer infrastructure services matching user's application requirements. In case the match occurs the broker deploys the application with the Cloud that was suggested by the CIS.

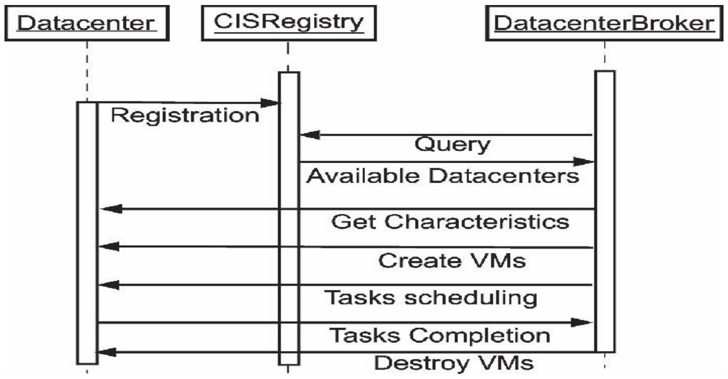


Figure. 2. State Diagram for CloudSim

The communication flow described so far relates to the basic flow in a simulated experiment. A variation in the flow of communication depends upon polices applicable to Virtual Machine or Data Center.[7]

III. PARTICLE SWARM OPTIMIZATION (PSO)

Particle Swarm Optimization is an Artificial Intelligence based techniques that may be applied to the problem domain where one have to find the approximate solutions for hellacious max-min numeric problems. In computer science **particle swarm optimization** (**PSO**) is a arithmetical method that give grade solutions while optimizes a problem by repeatedly trying to improve a candidate solution. PSO evaluates a problem by having a number of candidate solutions, which are known as particles, and moving these particles around in the search-space fitting to simple mathematical formulae over the particle's position and velocity. The movement of each particle is affected by its local best known position and is also guided as to the best known positions in the search-space, these are updated as better positions and then they are found by other particles. This is expected to move the best particle termed as swarm toward the best solutions.[1]

[James Kennedy and Russell Eberhart,1995] introduced method for optimization of continuous nonlinear function. Authors presented their work details using concepts from different areas. They covered social as well as animal behavior and hence studied bird flocking by Heppner, Grenander (1990), Reynolds (1987) and fish schooling by Wilson(1975).Their paper described human's behavior and preferred optimization in terms of multidimensional space and collision free.[8][1]

Their paper described PSO concept in terms of precursors and revised the various stages of its progression from social simulation to optimizer. The precursors were nearest neighbor velocity matching & craziness, cornfield vector, elimination of ancillary variable like craziness and considered flock behaved as swarm, acceleration by Distance. Authors selected the label to define the optimization concept was Particle Swarm. For optimizing continuous nonlinear function, authors presented Particle Swarm Algorithm in simple manner with its optimizer. They performed experiments by using Schaffer's f6 function. Over a series of 10 training sessions, the particle swarm optimizer required an average of 284 epochs. The particle swarm optimizer was able to train the network so as to achieve 92 percent correct.[8]

PSO can search for extremely large search space of candidate solutions without making any assumptions or very few assumptions about optimization of the problem. However, PSO do not guarantee an optimal solution is ever found. PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods. PSO can be used for optimizing the problems such as partially irregular, noisy, change over time, etc. Once the PSO is based on a simple concept, the social behavior of particles in the swarm, it has attracted many researchers' attention and has been applied with success to complex engineering problems, mainly in nonlinear function minimization, optimal capacitor placement in distribution systems, shape optimization, dynamic systems and game theory, constrained optimization, multiobjective optimization problems, electromagnetic, control systems, planning of electrical systems, and others.

PSO Algorithm

Formally, for PSO algorithm let $f\colon \mathbb{R}^n \rightarrow \mathbb{R}$ is the cost function which must be minimized. The function takes a particle as argument in the form of a vector of real numbers and produces a real number as output which indicates the objective function value of the given particle. For initialization, suppose that the gradient of f is not known. Now for processing, the goal is to find a solution **a** for which $f(\mathbf{a}) \le f(\mathbf{b})$ for all **b** in the search-space, which would mean **a** is the global minimum. Maximization can be performed by considering the function $h = -f$ instead.

Assume S be the number of particles in the swarm, and for each having a position $\mathbf{x}_i \in \mathbb{R}^n$ in the search-space and a velocity $\mathbf{v}_i \in \mathbb{R}^n$. Let $\mathbf{p}_i$ be the best known position of particle i and let **g** be the best known position of the entire swarm in the search space. A basic PSO algorithm is then:

- For each particle $i = 1, ..., S$ do:
    - Initialize the particle's position with a uniformly distributed random vector: $\mathbf{x}_i \sim U(\mathbf{b}_{lo}, \mathbf{b}_{up})$, where $\mathbf{b}_{lo}$ and $\mathbf{b}_{up}$ are the lower and upper boundaries of the search-space.
    - Initialize the particle's best known position to its initial position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
    - If $(f(\mathbf{p}_i) < f(\mathbf{g}))$ update the swarm's best known position: $\mathbf{g} \leftarrow \mathbf{p}_i$
    - Initialize the particle's velocity: $\mathbf{v}_i \sim U(-|\mathbf{b}_{up}-\mathbf{b}_{lo}|, |\mathbf{b}_{up}-\mathbf{b}_{lo}|)$
- Until a termination criterion is met (e.g. number of iterations performed, or a solution with adequate objective function value is found), repeat:
    - For each particle $i = 1, ..., S$ do:
        - For each dimension $d = 1, ..., n$ do:
            - Pick random numbers: $r_p, r_g \sim U(0,1)$
            - Update the particle's velocity: $\mathbf{v}_{i,d} \leftarrow \omega\, \mathbf{v}_{i,d} + \varphi_p\, r_p\, (\mathbf{p}_{i,d}-\mathbf{x}_{i,d}) + \varphi_g\, r_g\, (\mathbf{g}_d-\mathbf{x}_{i,d})$
        - Update the particle's position: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$
        - If $(f(\mathbf{x}_i) < f(\mathbf{p}_i))$ do:
            - Update the particle's best known position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
            - If $(f(\mathbf{p}_i) < f(\mathbf{g}))$ update the swarm's best known position: $\mathbf{g} \leftarrow \mathbf{p}_i$
- Now **g** holds the best found solution.

The parameters ω, φ_p, and φ_g are selected by the interpreter and control the behavior and efficiency of the PSO method.[1]

Related Work in PSO

[Zhao Yongyi, et-at, 2010] presented a new algorithm PSOB for Load balancing on Multidimensional Network Services which provided effective & efficient Load Balancing of Network. In Mathematical modeling for network values of service load, costs has been considered for the network and used for the calculation of an objective function. Inertia weight has been adjusted after then it used in load balancing process as significant parameter. The inertia weight considered critical for the PSOB quick convergence and hence it must be adjusted appropriately by maintaining fitness values of the particles using fitness function. This used in the load balancing algorithm for further processing. Depends on fitness value and iteration, results determined by the algorithm, showed that PSOB achieved fast convergence which had inertia weight that played a major role in the optimization of load balancing process.[3]

Based on the loading conditions, PSOB tried to balance load of network communication flows by auto adaptive operation through which adjusted services load on the network nodes. Mathematical modeling has been presented for the network which calculated an objective function. Inertia weight was adjusted which then used in load balancing process as significant parameter. PSOB algorithm was executed in standard C++ program using VC & GCC compiler. The data was produced by random function of normal distribution in MATLAB. Iteration number has lager value than fitness and the fitness began to wobble within 160iteration and converged at last, through which

PSO's fast convergence proved. The iteration number was larger than the value which has been estimated. When relation between the value of εη and Iterations plotted into a graph, it showed that during iteration 10to150 fitness was changed accordingly but it began to wobble within 160 iterations and converge at last through which better performance of PSOB achieved.[3]

Concrete congestion dispatching method and its technical study for PSO and the model experiment proved that the algorithm performed with better converging action and overarching adjustment.

[Li Li, et-at, 2010] woked on Multiobjective job shop scheduling problem and developed algorithm in which Traditional ACA combined with PSO used .This provide Feasible & effective approach for flexible job shop scheduling problem. Their algorithm made full use the parameters of fast convergence of PSO and positive feedback of ACA determined by the same parameters, which were repeatedly updated by performing the comparison between old and new fitness values so that the solution would be improved and strengthen the search capability for optimal solution and quick convergence of algorithm. The finalized algorithm was capable to solve multi objective FJSSP in very effective and efficient manner. [4]

First scheduling had been performed on the basis of transition possibilities and state transition rule, object function could be calculated. Local search was carried out and using that global updating had been carried out according to best scheduling. Optimum scheduling could be obtained for each subset and this procedure had been repeated until maximal subsets could be met. Experiments carried out tests with problem 4×5 with 12 operations and problem 8×8 with 27 operations. The value of object function was 14.8 for problem 4×5 and 26.0 for problem 8×8 which was lesser than the values calculated by applying other algorithms.[4]

Related work in PSO with Cloud Model

[Changming Zhu, et-at, 2012] developed a model which worked on problem of premature or slow convergence rate for cloud model. Cloud generator had two parts-forward generator and backward generator. Forward generator used Ex, En and He to generate cloud drop (x, μ) where x denoted quantity value and μ denoted membership degree of x. Backward generator used to convert quantity concept into quality concept i.e. (x, μ) into Ex, En, He. This cloud model was used in crossover operation of DE algorithm to avoid getting into local optimum. The resultant algorithm was CMDE in which first fitness value calculated, after then mutation and crossover with cloud model performed. Then selection made between original individual and excellent individual. Algorithm run repeatedly until termination condition gave optimal output in result. For solving slow convergence speed of DE algorithm, cloud model applied in the crossover operation of algorithm which gave better convergence rate and reliable result. For 9 benchmark functions, algorithm made 50 run and the result was compared to other algorithms. Without prior knowledge of user interaction, presented algorithm gave better performance.

Cross over strategy implied into cloud based model with self adaptive rule for parameter settings, showed better convergence performance. CMDE gave better performance in terms of convergence rate and reliability on both uni-modal and multimodal benchmark functions.[5]

[Ying Gao, et-at, 2010] presented an algorithm for performance optimization on cloud model. The specific question answered was global information about search space and local information of solutions provided by a single methodology which had PSO and CMBOA approaches together.

Cognitive population was used to estimate good solution regions and generate new particles in the search space. Three characteristics expected value, entropy and super entropy for the cognitive population was made via backward cloud generator. After then cloud drop used to generate same characteristics using forward cloud generator. New generation of Population generated through PSO particles and cloud particles. For improving search ability, cloud model applied with PSO which gave better result with weight factor.

The proposed algorithm was better with weight factor and remarkable standard deviation for all test functions. Presented algorithm was effective and had stronger global search ability than original version of PSO. local information from PSO particles and global information from cloud particles were used together to guide the overall search.[6]

[Yan Gao-wei, et-at, 2012] presented an algorithm based on cloud model for solving numerical optimization problem. The novel numerical stochastic optimization was done by algorithm which used the generation behavior, move behavior and spread behavior of cloud in a simple way.

Search space was disjoint into several separate regions. Cloud generation, Cloud move and cloud spread were the steps for the algorithm. Cloud generated for only those regions whose humidity value was higher than certain threshold and moved from higher air pressure value to lower air pressure value. The algorithm updated the humidity values and air pressure values for all regions every time after each step.

Four test functions schaffer, needle in haystack, yang and rastingin applied for PSO, GA and ACMO respectively. In case of yang function's graph at iteration 20, fitness value was 1 for ACMO, but for PSO fitness value was 1 at iteration 60 and in case of GA fitness value was 0.8 at iteration 85.So it proved that ACMO worked better in convergence process when it compared with PSO and GA. ACMO algorithm had a good evolutionary ability by preventing premature convergence rate and it could escape from local optimum as compared to GA or PSO. It found that the presented ACMO algorithm could find optimum 100% for all functions except needle in haystack function.[7]

IV. PSO RUNS IN CLOUDSIM

The basic algorithm which we used to be executed in the cloudSim has simple concept without affecting the code of basic cloudSim's classes like DataCenter, VirtualMachine, DataCenterScheduler etc. For implementing the PSO we select 5 VMs and generate 2 DataCenteres with 2 Hosts.The experiment was conducted on Fujitsu machine with Intel Core i3 processor with 4 GB RAM in NetBeans IDE having version 7.3 with jdk 1.7. The algorithm which we used is implemented in such system but we have to do more work in the further directions.

```
Algorithm: PSO-Particle Swarm Algorithm
1. Initialize gBest, gBestTest and epoch with 0s.
2. Call initialize() [for particle generation]
3. Repeat while(false)
4.        if epoch<Max_epoch
5.                do for i->0 to Max_epoch
                          Particle.get(i)
6.                        if testProblem(i)==Target
                                  Set condition true for while
7.                gBestTest=minimum();[minimum in the particle]
8.                particle.get(gBest)
9.                if Target-testProblem(gBestTest)< Target-testProblem(gBest)
                          gBest=gBestTest
10. a)            getVelocity(gBest) [retrieve velocity of gBest particle]
    b)            updateparticles(gBest) [update particles with effect to gBest]
    c)            epoch++ [make an increment in epoch's current value]
11.       else set condition true for while
12. end of function.
```

The language used to explain the PSO follows from the analogy of particles in a swarm. These key terms are as following-
pbest (personal best): the position in parameter space of the best fitness returned for a specific particle;
gbest (global best): the position in parameter space of the best fitness returned for the entire swarm;
V_{max}: the maximum velocity value allowed in a given direction.

The main function of our code in which gBest and gBestTest are used as variables is PSOAlgorithm. To compare the value of particle with its local value with the global value we used these names. First we generate all the particles to be used in the experiment using initialize() function. Then we run a loop with condition false to make checking of epoch with Max_epoch.We retrieve one by one particle id and make comparison between gBest and gBestTest value by keeping the Target in mind and update the others as per the matching of the criteria. In the end we get the actual value of particle which can be match with the target.

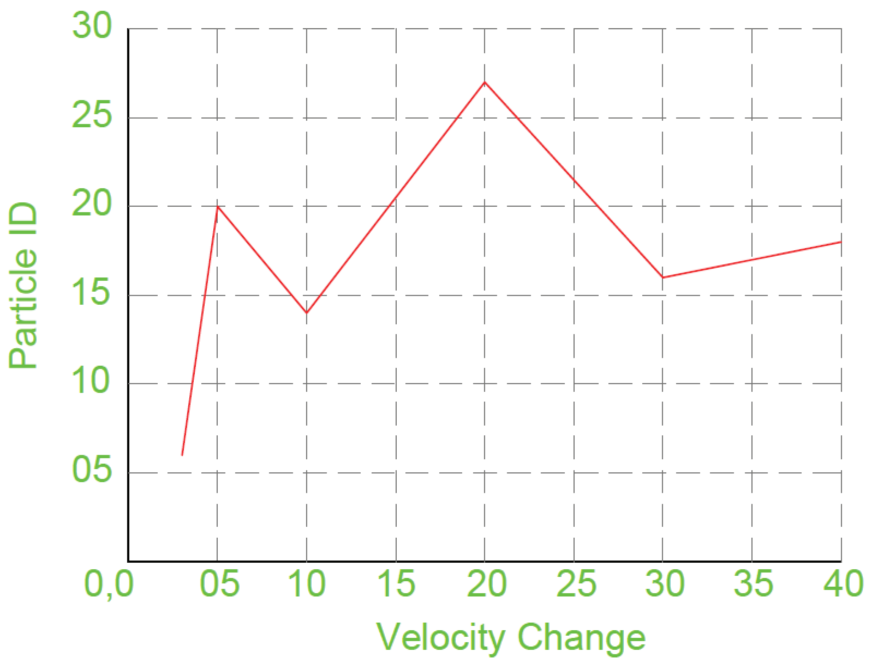
For fix number of particles i.e. 35 and fix number of epochs i.e. 250 we make changes in velocity and check the results. Following is the result analysis in tabular form.

Sr No.	Velocity Change allowed	Target	Particle id
1.	0	80	Error
2.	2	80	6
3.	5	80	20
4.	10	80	14
5.	20	80	27
6.	30	80	16
7.	40	80	18

TABLE 2
RESULT ANALYSIS for PSO in CloudSim

The result shows that when we make changes in the number of velocity to be changed than for the same target the particle id would also be changed in the same cloud environment having 2 host, 5 VMs and 2 DataCenters.

When we plot the graph between velocity change and Particle id than we found that for particle id 27 , the velocity changes maximum. When we do not apply any change in the velocity than there must not any change in the result or generally it gives an error. X-axis of graph shows the change in velocity and the Y-axis shows the particle id.



V. CONCLUSION

First section covered the concept of cloud computing and cloudSim; second presents the study of PSO and literature survey done in that area was given in third; the last section covers the main idea of the paper i.e. implementation of PSO in CloudSim. We presented an analytical study based on PSO as well as cloudSim. Particle Swarm Optimization algorithm has lots of features like good convergence rate, less expensive, easy to apply in different scenario and simple to implement. When cloud model applied with PSO then, the efficiency of result enhances and sometime it gave 100% results also. We presented the result of implementation of PSO in cloudSim which is the basic idea that we can execute this algorithm in any new environment or any new simulator. The result shows that when we fix the target than by changing in velocity we can get new particle id. Change in velocity has major concern as particle id directly changes.

We can propose variants of PSO to be executed in CloudSim and in near future we can also assume to perform the search in cloudlets rather than doing the search in particles using particle simulation algorithm. This work has lots of new scopes to be achieved and we just put the basic by implemented the standard particle swarm algorithm.

REFERENCES

[1] Kavita Bhatt, Mahesh Bundele, Review Paper on PSO in workflow scheduling and Cloud Model enhancing Search mechanism in Cloud Computing, IJIET-International Journal of innovations in engineering and technology, Vol.2 issue 3, June 2013.

[2] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, Rajkumar Buyya, A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments, 24th IEEE International Conference on Advanced Information Networking and Applications, 2010, ISBN: 978-0-7695-4018-4,Pg 400.

[3] Zhao Yongyi, Xia Shengxian, Research on Load Balancing for Multidimensional Network Services Based on Particle Swarm Optimization Algorithm, Third International Conference on Intelligent Networks and Intelligent Systems,2010, ISBN: 978-0-7695-4249-2, Pg 411.

[4] Li Li, Wang Keqi, Zhou Chunnan, An Improved Ant Colony Algorithm Combined with Particle Swarm Optimization Algorithm for Multi-objective Flexible Job Shop Scheduling Problem, International Conference on Machine Vision and Human-machine Interface,2010, ISBN:978-1-4244-6595-8, Pg 88.

[5] Changming Zhu, Jun Ni, Cloud Model-Based Differential Evolution Algorithm for Optimization Problems, Sixth International Conference on Internet Computing for Science and Engineering,2012, ISBN:978-1-4673-1683-5,Pg 55.

[6] Ying Gao, Xiao Hu, Huiliang Liu, Fufang Li, Cloud Estimation of Distribution Particle Swarm Optimizer, Fourth International Conference on Genetic and Evolutionary Computing, 2010,ISBN: 978-1-4244-8891-9,Pg 14.

[7] Yan Gao-wei,Hao Zhanju, A Novel Atmosphere Clouds Model Optimization Algorithm, International Conference on Computing, Measurement, Control and Sensor Network,2012,ISBN: 978-1-4673-2033-7,Pg 217.

[8] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942-1948, IEEE Press, Piscataway, NJ, 1995.

[9] Cloud Confusion Amongst IT Professionals. Version One (June 6, 2011).

[10] http://www.wikinvest.com/concept/Cloud_Computing

[11] http://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031?page=0,2

[12] White paper on Cloud Computing in the Public Sector by Cisco.

[13] White paper on Cicso Cloud Computing: Data Center Strategy, Architecture and Solutions, by Cisco, 2009.