A PSO based VM Resource Scheduling Model for Cloud Computing

Dinesh Kumar
School of Computer & Systems Sciences
Jawaharlal Nehru University
New Delhi, India
dineshyadav2571@gmail.com

Zahid Raza
School of Computer & Systems Sciences
Jawaharlal Nehru University
New Delhi, India
zahidraza@mail.jnu.ac.in

***Abstract*— Cloud computing, a new paradigm for utility computing, has brought a revolutionary change in the IT industry by enabling the elastic on demand resource provisioning of computing resources. As cloud infrastructure heavily relies on virtualization technology, there is a need for an efficient and effective virtual machine scheduling strategy. Virtual machine scheduling problem can be defined as an allocation of a set of virtual machines (VMs) to a set of physical machines (PMs). The proposed work focuses on PSO based VM scheduling strategy for VM placement in cloud infrastructure. The strategy focuses on efficient VM allocation to physical servers in order to minimize the total resource wastage and the number of servers used. Simulation experiments were conducted to observe the allocation of VMs to the servers and to evaluate the proposed algorithm with respect to performance and scalability. The results are compared with Best-Fit, First-Fit and Worst-Fit placement strategies. Simulation study conducted to evaluate the performance reveals the effectiveness of the model.**

Keywords—Cloud Computing, Virtual Machine Placement, Particle Swarm Optimization, Virtualized Data Center, Multi-dimensional Bin Packing

I. INTRODUCTION

Cloud Computing refers to the parallel and distributed computing paradigm that involves delivering hosted services through internet on a *pay per use* model. In its various forms and scenario, the main goal of Cloud Computing is to deliver storage, network, servers, computing or their combination "as a service" [1,2]. As Cloud infrastructure depends on virtualization technology, wider adoption of cloud will increase more VM requests and hence an increase in the virtualized infrastructure and therefore need an efficient VM scheduling strategy. Many benefits and desirable characteristics of cloud computing such as scalability, load balancing, elasticity etc. are possible due to virtualization technology. VM scheduling problem is proven to be an NP Hard Problem, implying an exact solution can't be found. Rather suboptimal or near optimal solutions can be found with the help of some soft computing approaches [3]. The strategy presented in this paper focuses on efficient VM allocation to physical servers in order to minimize the total resource wastage and number of servers used.

The remainder of the paper is organized as follows. In Section II, related work on VM placement optimization is discussed. Section III discusses the resource wastage model and problem statement and its formulation. In Section IV, resource wastage aware VM placement approach is discussed while presenting the data structures and notations used. Section V discussed the resource wastage aware PSO based VM placement algorithm. The computational results are presented in Section VI with the work concluding in Section VII.

II. RELATED WORK

VM placement is the process of mapping of virtual machines to physical machines. As the cloud infrastructure is wholly virtual, VM placement becomes a core issue in cloud system. A lot of research work has been devoted to solve the VM placement problem using various strategies. VM placement problem is often formulated as a variant of bin-packing problem which is an NP-Hard problem [4]. Accordingly, a number of models with varying assumptions and constraints have been proposed in the Literature. In [5], VM placement problem is modeled as a multidimensional bin packing problem where servers are bins with each resource(CPU, memory, network, storage etc.) being one dimension of bin and VM are objects that are to be inserted into bins. The algorithm proposed in [6] is based on first fit approximation approach for the bin packing problem. The heuristic approach proposed in [7] is derived from First fit decreasing algorithm. The framework presented in [8] solves the VM placement problem using the constraint programming paradigm. Linear programming is another traditional analytical approach to solve VM placement problem. The approach proposed in [9] allocates the virtual machines on physical machines to minimize the number of nodes used which uses the linear and quadratic programming strategy. Another approach proposed to solve this problem is presented in [10] where an online self-reconfiguration approach is presented for allocation of VMs using genetic algorithm. The heuristic approach presented in [11] gets derived from the Best-Fit decreasing algorithm.

III. RESOURCE WASTAGE MODEL

In a cloud environment, there are large data centers having a pool of server nodes. As the infrastructure is fully virtualized, all the applications run on Virtual machines (VMs) and these VMs are deployed on physical servers. In current scenario, only two dimensions of a VM are considered i.e. CPU capacity and memory size. The CPU utilization of a server is estimated as the sum of CPU utilization of all the VMs that are running on that server. The memory utilization can also be calculated in the same manner. For example if there are two VMs running on a server with (35%, 20%) and (25%, 50%) utilizations where (35%, 20%) represents the pair of CPU and memory requests of first VM and (25%, 50%)

978-1-4799-6023-1/15 $31.00 © 2015 IEEE
DOI 10.1109/CICT.2015.35

represents the pair of CPU and memory requests of the second VM then the total CPU and memory utilization of that server would be (60%, 70%) being a sum of the vectors [12].

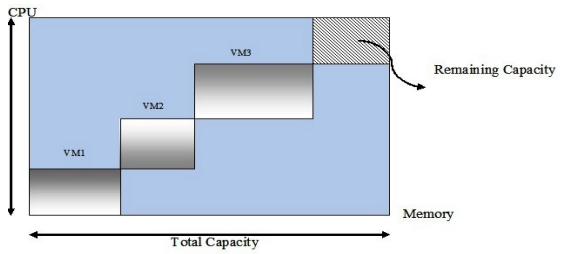

Figure 1: An Example of Resources Allocated to Three VMs Running on a Single Server.

The size of the VM can be represented as a d-dimensional vector in which each dimension corresponds to one type of the requested resource (example CPU, memory, storage). For example, the diagram shown in Figure 1 illustrates the allocation of three VMs to a single server where each VM has only two dimensions. As can be seen, after deploying the VMs, some amount of resources remain unutilized or go wasted in the allocation process. Our main objective of the VM placement algorithm is to minimize the resource wastage. Unutilized resource on each server may vary largely with different VM placement solutions. Remaining resource should be balanced along different dimensions because unbalanced unutilized resource may prevent any further VM placement thus wasting computing resources.

A. *Resource Wastage Modelling*

The remaining resources available on each server depend upon the VM placement strategy used while deploying the VMs on physical servers. Therefore, it is always desired that the VMs should be deployed in such a way so as to minimize the total resource wastage. To fully utilize multidimensional resources, following equation is used to calculate the potential cost of the wasted resources [12]:

$$W_j = \frac{|R^p{}_j - R^m{}_j| + \varepsilon}{U^p{}_j + U^m{}_j} \tag{1}$$

Here, W_j represent the resource wastage of jth server, $U^p{}_j$ and $U^m{}_j$ represent normalized CPU and memory utilization i.e. ratio of the used resource to the total resource. $R^p{}_j$ and $R^m{}_j$ represent the normalized remaining CPU and memory resource i.e. ratio of remaining resource to the total resource. ε is a small positive number and its value is set to 0.000001. The key idea behind the above equation is to balance the remaining resources along different dimensions and to make the effective use of the resources in all dimensions.

B. *Problem Formulation*

Suppose that there are *n* VMs (applications) i $\in$ I that are to be placed on *m* servers j $\in$ J. Let us assume that none of the VMs requires more resources that can be provided by any single server. Let $R^p{}_j$ and $R^m{}_j$ be the CPU and memory demand of ith VM, where $T^p{}_j$ and $T^m{}_j$ are the CPU and memory thresholds of jth server. A binary variable x_{ij} indicates that ith VM is running on the jth server. The variable y_j indicates whether the server is turned on or off. Since, the objective is to minimize the total resource wastage; the placement problem can be formulated as:

$$Minimize \sum_{j=1}^{m} W_j = \sum_{j=1}^{m} y_j \times \frac{|((T^p{}_j - \sum_{i=1}^{n}(x_{ij}.R^p{}_i))/T^p{}_j) - ((T^m{}_j - \sum_{i=1}^{n}(x_{ij}.R^m{}_i))/T^m{}_j)| + \varepsilon}{\sum_{i=1}^{n}((x_{ij}.R^p{}_i)/T^p{}_j) + \sum_{i=1}^{n}((x_{ij}.R^m{}_i)/T^m{}_j)} \tag{2}$$

Subject to:

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall i \in I \tag{3}$$

$$\sum_{i=1}^{n} x_{ij}.R^p{}_i \le T^p{}_j.y_j \qquad \forall j \in J \tag{4}$$

$$\sum_{i=1}^{n} x_{ij}.R^m{}_i \le T^m{}_j.y_j \qquad \forall j \in J \tag{5}$$

$$y_j, x_{ij} \in \{0,1\} \qquad \forall j \in J \text{ and } \forall i \in I \tag{6}$$

IV. RESOURCE WASTAGE AWARE PSO BASED VM PLACEMENT APPROACH

Particle swarm optimization is a stochastic, population based meta heuristic computational method and inspired by the social behavior in human beings and animals especially fish schooling or bird flocking [13]. The credit for PSO goes to Kennedy, Eberhart and Shi [14] and was first intended to simulate the social behavior in animals [15]. Initially, all particles of swarm are randomly initialized in multi dimensional search space. The dimensionality of the search space depends upon the problem that needs to be solved. The particles perform flights in the multi dimensional search space with their adjusted velocities and change their positions subsequently. Each particle is represented by two things, its current position and velocity. The position of a particle represents a candidate solution to the problem. Each particle changes its position by moving towards better position by changing its velocity which is governed by some rules. These rules are originally inspired by behavioral models of bird flocking or school flocking [16]. In PSO, each particle stores two values: (1) its own best position (localBest) and (2) the best position obtained by the whole swarm (globalBest). These two values are used for updating the velocity of a particle and is presented as equation (7) [17].

$$V_{id}(t+1) = V_{id}(t) + C_1 r_{1d}(P_{id} - X_{id}) + C_2 r_{2d}(P_{gd} - X_{id}) \tag{7}$$

Where C_1 and C_2 are two positive numbers called acceleration coefficients and r_{1d} and r_{2d} are two uniformly distributed numbers in interval [0, 1]. P_{id} and P_{gd} denotes the localBest and globalBest positions of a particle. After

changing their velocity, the position of a particle is calculated by the following equation [17].

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1) \quad (8)$$

A. PSO for VM Scheduling

In this work, a PSO based VM Scheduler is used for placement of virtual machines on the physical servers. As explained earlier, PSO has many advantages over other Meta heuristics techniques and has some interesting properties which are very useful for solving some NP Hard and NP Complete problems being easy to implement and having lesser parameters to adjust. Before applying PSO to VM scheduling problem, it needs certain modifications and improvements because VM scheduling problem is a discrete optimization problem but a basic PSO is suitable for solving only continuous problems. To solve the virtual machine placement problem, the encoding scheme, position update strategy and velocity update strategy of PSO must be adjusted. Therefore, operations and parameters of the standard PSO have to be redefined so that it is capable of solving VM scheduling problem.

1) Encoding Scheme: A two-dimensional encoding scheme is quite useful for solving the VM placement problem. The first dimension of a particle is an n-bit vector where n is the number of physical servers and the second dimension contains the set of subsets comprising the VMs to be placed on those servers [18]. In the first dimension, every bit is associated with a server. If the bit is '1' it means that the server is active and at least one VM is running on that server. If the bit is '0' then it means that corresponding server is idle and not in use.

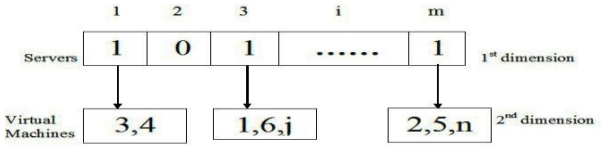

Figure 2: Two – Dimensional Encoding Scheme [18]

2) Parameters and Operators of PSO Used: Modifications of standard PSO can be done by redefining its parameters and operators. The parameter and operators of the PSO used in the work are presented below [18]:

a) Particle Position: The particle position X^t_i is as an n - bit vector where $X^t_i = (X^t_{i1}, X^t_{i2}, \dots, X^t_{in})$ where n is the number of physical servers in cluster. If j^{th} bit is 1, it means that j^{th} server is in use and at least one VM is running on it. If the bit is 0, then corresponding server is idle and not in use.

b) Particle Velocity: The particle velocity V^t_i is an n-bit vector where $V^t_i = (V^t_{i1}, V^t_{i2}, \dots, V^t_{in})$. Here, the velocity of a particle represents the adjustment decision of the virtual machine placement solution. Every bit in velocity vector is 0 or 1. If the bit is 0, then the corresponding server and virtual machines running on that server must be reevaluated. If the bit value is 1, then there will be no change and the corresponding server is not evaluated.

c) Subtraction operator: The subtraction operator is used to calculate the difference between the two feasible VM placement solutions. The symbol Θ can be used to represent the subtraction operator. For given two solutions X^t_i and X^t_j, if the corresponding bits of both solutions are same, then the bit value in the result will be 1, otherwise 0. For example $(1,0,1,0) \Theta (1,1,0,0) = (1,0,0,1)$.

d) Addition Operator: The addition operator is used to represent the particle's velocity update. The velocity update of a particle will be caused by three factors viz. current velocity inertia, local best position and global best position. Here, the symbol $\oplus$ is used to represent the addition operator. If there are n physical servers present in the cluster, then $(P_1V'_1 \oplus P_2V'_2 \oplus \dots P_nV'_n)$ denotes that a particle updates its velocity by using V^t_1 with a probability P_1, V^t_2 with P_2, using V^t_n with a probability P_n and so on. For example, $0.4(1,1,0,0) \oplus 0.6(1,0,1,0) = (1,\#,\#,0)$. The probability that the value of second bit is equal to 1 is 0.4, and the probability that value is equal to 0 is 0.6. Thus the update of the particle velocity depends upon these uncertain values. These uncertain values are calculated by finding the three inertia weight coefficient values, P_{1i}, P_{2i}, and P_{3i}. Values of these coefficients can be obtained by the following equations [18].

$$P_{1i} = \frac{1/f(X^t_i)}{1/f(X^t_i) + 1/f(X^t_{localBesti}(t)) + 1/f(X^t_{globalBest}(t))} \quad (9)$$

$$P_{2i} = \frac{1/f(X^t_{localBesti}(t))}{1/f(X^t_i) + 1/f(X^t_{localBesti}(t)) + 1/f(X^t_{globalBest}(t))} \quad (10)$$

$$P_{3i} = \frac{1/f(X^t_{globalBest}(t))}{1/f(X^t_i) + 1/f(X^t_{localBesti}(t)) + 1/f(X^t_{globalBest}(t))} \quad (11)$$

In the above equation $f(X^t_i)$ denotes the fitness of solution represented by the i^{th} particle with its current position X^t_i and is discussed in the next section. The fitness of a VM placement solution is the total resource wastage of all servers in a cluster. Higher the resource wastage of the feasible solution, the greater will be its fitness. Hence, the solution having greater fitness will have the smaller value of inertia weight coefficient. Accordingly, selection probability of the solution that has greater fitness (i.e. more resource wastage) will be lower as compared to the other with lesser fitness. After finding the inertia weight coefficients, the uncertain bit values can be obtained by the following rules:

The uncertain bit value = *q1*, if $rand <= P_{1i}$
The uncertain bit value = *q2*, if $P_{1i} < rand <= P_{2i}$
The uncertain bit value = *q3*, if $P_{2i} < rand <= P_{3i}$

Here *rand* is a random number generated in the range 0.0 to 1.0, *q1* is the corresponding bit value before updating the particle velocity, *q2* is the corresponding bit value of a particle localBest velocity vector and *q3* is the corresponding bit value of the particle's global best velocity vector.

e) Multiplication Operator: The multiplication operator is used to update the particle position. Suppose $\otimes$ is the multiplication operator. $X^t_i \otimes V^{t+1}_i$ represents the current particle position update based on the velocity vector V^{t+1}_i. If

the bit value of the velocity vector is '1', then the corresponding bit of the position vector is not changed. If the corresponding bit of the velocity vector is '0' then it shall be adjusted. For example $(0,0,1,1) \otimes (1,0,0,1)$ are the position and velocity vector then according to the position update strategy, the second and third bit of position vector shall be changed. It means that the VMs running on second and third server should be re evaluated and checked whether it is an optimal way of placing the VMs or not. The velocity and position update equations are given below:

$$V^{t+1}{}_i = P_1 V^t{}_i \oplus P_2(X_{localBesti}(t) \Theta X^t{}_i) \oplus P_3(X_{globalBest}(t) \Theta X^t{}_i) \quad (12)$$

$$X^{t+1}{}_i = X^t{}_i \otimes V^{t+1}{}_i \quad (13)$$

3) Particle Position Update Strategy: After updating the velocity of a particle, its position is updated. As explained earlier, if the corresponding bit value in velocity vector is '0', its position is adjusted otherwise it remains the same. The position update should be in such a way that the particles can search the whole search space effectively and efficiently. To obtain the global optimal VM placement solution, we adopt a resource wastage aware local fitness first strategy. Here, every bit in the first dimension is called as local position. For this local position, local fitness is defined as the sum of CPU and memory utilization of all the VMs that are running on that local server. The local fitness of j^{th} server can be calculated by equation given below:

$$f_{local\_current_j} = \frac{\sum_{i=1}^{n} x_{ij}.R^p{}_i}{T^p{}_{current_j}} + \frac{\sum_{i=1}^{n} x_{ij}.R^m{}_i}{T^m{}_{current_j}} \quad (14)$$

Where $T^p{}_{current_j}$ and $T^m{}_{current_j}$ denotes the CPU and memory threshold of j^{th} local server. $R^p{}_i$ and $R^m{}_i$ denotes the CPU and memory requirements of i^{th} VM. The binary variable x_{ij} tells whether the i^{th} VM is running on j^{th} server or not. The value of x_{ij} is 1 if the i^{th} VM is allocated to j^{th} server, otherwise 0.

Similarly, the local fitness of j^{th} server i.e. j^{th} local position of localBest and globalBest positions can be calculated as:

$$f_{local\_localBest_j} = \frac{\sum_{i=1}^{n} x_{ij}.R^p{}_i}{T^p{}_{localBest_j}} + \frac{\sum_{i=1}^{n} x_{ij}.R^m{}_i}{T^m{}_{localBest_j}} \quad (15)$$

$$f_{local\_globalBest_j} = \frac{\sum_{i=1}^{n} x_{ij}.R^p{}_i}{T^p{}_{globalBest_j}} + \frac{\sum_{i=1}^{n} x_{ij}.R^m{}_i}{T^m{}_{globalBest_j}} \quad (16)$$

After finding the local fitness values, the three probability values can be written as:

$$P_1 = \frac{f_{local\_current_j}}{f_{local\_current_j} + f_{local\_localBest_j} + f_{local\_globalBest_j}} \quad (17)$$

$$P_2 = \frac{f_{local\_localBest_j}}{f_{local\_current_j} + f_{local\_localBest_j} + f_{local\_globalBest_j}} \quad (18)$$

$$P_3 = \frac{f_{local\_globalBest_j}}{f_{local\_current_j} + f_{local\_localBest_j} + f_{local\_globalBest_j}} \quad (19)$$

As described above, if the j^{th} bit of particle current velocity is '0' then the j^{th} server is re evaluated. For updating the configuration of j^{th} server the following strategy is used.

- Generate a random number *rand* between 0 and 1.
- If *rand* < P1 then, there will be no change.
- If P1 < *rand* $\leq$ P2 then, replace the corresponding VMs in current solution with the VM set of j^{th} server of localBest solution. For example suppose 3^{th}, 4^{th} and 7^{th} VMs are running on j^{th} server in the currently obtained solution and 4^{th}, 5^{th} and 6^{th} VMs are in the VM set of localBest solution, then in that case, the 3^{th}, 4^{th} and 7^{th} VMs are replaced by 4^{th}, 5^{th} and 6^{th} VMs.
- If *rand* > P2, replace the corresponding VMs in the current solution with the VM set of j^{th} position of globalBest solution. For example suppose 9^{th}, 4^{th} and 6^{th} VMs are running on j^{th} server and 2^{th}, 11^{th} and 8^{th} VMs are in the VM set of globalBest solution, then in this case, the 9^{th}, 4^{th} and 6^{th} VMs are replaced by 2^{th}, 11^{th} and 8^{th} VM.

B. Data Structures Used in the Model: The following data structures are used to meet the requirements:

- Pop [number of Particles] [number of servers]: This matrix represents the whole population. The number of rows is equal to the number of particles and number of columns is equal to number of physical servers.
- PMList: This List stores the information of all physical servers, for example each node in list represents a single physical server with its ID, its CPU threshold and memory threshold.
- VMReqList: This list contains the information about each VM for example, its ID, its CPU requirements and its memory requirements.
- Particle: ArrayList is used as a data structure to store a particle. The length of the ArrayList is equal to number of servers. Each node in ArrayList contains another ArrayList which is used to represent the set of VMs that are running on corresponding server.
- LocalBest [number of particles] [number of servers]: Each particle stores its local Best position that has been found so far by itself. This matrix is used to store the localBest position of all particles.
- GlobalBest [number of servers]: ArrayList is used to represent the globalBest position. The length of the ArrayList is equal to number of servers.

V. THE ALGORITHM

The PSO based VM Placement algorithm schedules the virtual machines on different servers so that the total resource wastage can be minimized. The algorithm starts with random initialization of the particles i.e. random feasible solutions are generated by randomly allocating VMs to the servers. After

initialization, all particles traverse in the solution space and move towards global optimal solution in successive iterations. The various steps of the algorithm are presented below.

1. Input set of VM Requests with their associated resource demands and set of Hosts with their thresholds of resource utilizations as Input to the algorithm.
2. Set *P* as the swarm size i.e. number of particles. The program is run for a fix number of iterations *maxIter*.
3. Generate the initial population. A random strategy is adopted to generate initial feasible random solutions. The number of initial feasible solutions is *P* i.e. Number of particles. Each particle represents a complete feasible VM placement solution. Accordingly, each particle is randomly initialized by generating a random feasible solution. After that, the particle velocities are initialized.
4. Find the localBest position values of all the particles by evaluating the fitness function. After finding the localBest, obtain the globalBest.
5. Repeat steps 6-10 for a number of iterations *maxIter*.
6. Repeat steps 7-9 for each particle.
7. Update the velocity of particle as per Equation (12).
8. Update the particle position as per Equation (13). After updating its position, the solution may or may not be feasible i.e. all the VMs may not be present in the solution. The missing or lost VMs are backfilled in the next step. In another case, a single VM could be running on two different servers. Therefore, these duplicate VMs are removed for achieving the feasible solution. These duplicated VMs are removed by simply turning off the servers on which these duplicated VMs are running. While performing the removing operation, the non-repetitive VMs are also deleted. So, these accidently deleted VMs are backfilled in next step. The backfilling operation adopts the best fit strategy. While re-inserting the VMs, first active servers are considered. If there is a possibility of placing the current VM on active servers, it is placed on an active server according to the best fit strategy, otherwise a new server is turned on and the current VM is placed on that new server.
9. Update the localBest position of particle by calculating its fitness value.
10. Update globalBest position by selecting the best position among the entire swarm.
11. Return the globalBest position which represents the global optimal VM placement solution.

VI. SIMULATION STUDY

Simulation experiments were conducted to observe the allocation of VMs to the servers and results are compared with Best-Fit, First-Fit and Worst-Fit strategy. The experiments were conducted on Intel(R) Core(TM) i5 processor @ 2.27GB using eclipse platform version Kepler service Release 2. Problem instances were randomly generated with the above configuration where the instances are the resource requirements of VMs. We consider only two dimensions of a VM i.e. CPU and memory and the demand set of all VMs containing their corresponding CPU and memory utilization requirements are generated randomly. In each case, the numbers of servers are equal to the number of VMs for supporting the worst case scenario which corresponds to only one VM is running on a single server.

TABLE I. PARAMETERS USED

S.No.	Parameter	Notation Used	Range
1	Number of VM Requests	N	100-500
2	Number of Servers	M	100-500
3	Number of iterations	MaxIter	100
4	Population size	Pop	500-2500
5	Number of independent run	Count	10

The experiment is performed for 100 generations. The particle size has been taken 1000 for 200 VMs and 200 servers. 20 types of servers are used in the experiment with CPU and memory threshold varying from 0.70 to 0.90. All the VM Requests are generated randomly. The algorithm for input generation is as follows [12]:

- For i = 1 to N do
- R^p_i = rand(2**MIPS_Ref*);
- R^m_i = rand(*mem_Ref*);
- *r* = rand(1.0);
- If (*r* < *probability* ^ $R^p_i \geq MIPS\_Ref$) ∨ (*r* ≥ *probability* ^ $R^p_i < MIPS\_Ref$) then
- $R^m_i = R^m_i + mem\_Ref$;
- End If
- End For

Where *MIPS_Ref* and *mem_Ref* represent the reference CPU utilization and reference memory utilization and the *probability* is a reference value. The correlation of CPU and memory utilizations can be controlled to some extent by varying *probability* value. In our experiment, *probability* value is 1 where as *MIPS_Ref* and *mem_Ref* are set to 0.35.

The experiment study is divided into three case studies corresponding to evaluation of the resource wastage when number of VMs is varied as Case I, number of active servers used by varying the number of VMs as Case II and finally the behavior of PSO in terms of the optimization towards minimizing the resource wastage over generations or time as Case III. Many sets of experiments were conducted and the results conform to the ones reported in the following discussion being on the similar line. In the Best-Fit (BF) strategy, first a set of servers on which a particular VM can be deployed is obtained. After that, the server with the minimum resource wastage is selected for that VM. The same strategy is followed for all the VMs. In the case of First-Fit (FF) strategy, the search starts from the first server and the VM is allocated to the first server it encounters which is large enough to satisfy the request. In Worst-Fit (WF) strategy, the scenario is opposite to the Best-Fit strategy i.e. the server with the maximum resource wastage is selected for a VM.

Case I: Resource Wastage v/s Number of VMs:

Figure 3 shows the resource wastage v/s number of VM requests. The results are compared with the Best-Fit, First-Fit and Worst-Fit strategy. As the number of VM requests increases, total resource wastage will also increases. The graph clearly shows that PSO based VM Scheduler perform the best among all approaches used. Highest resource wastage is obtained in the case of Worst-Fit, while Best-Fit performs better than First-Fit and Worst-Fit. Worst performance is observed in the case of Worst-Fit with the performance deteriorating with the increase in the VMs and servers. In sharp contrast, the PSO based approach ensures minimum and significantly lower resource wastage for all cases.

The results show that the performance of PSO based approach is the best among others resulting in the minimum resource wastage. The fundamental reason lies in that FF, BF, and WF are simple greedy approaches and they lack global information. Therefore, local optimal solutions are obtained with these approaches. In the PSO based approach, a vast solution space is traversed and an optimal allocation for each VM is attained using the resource wastage aware local fitness strategy.

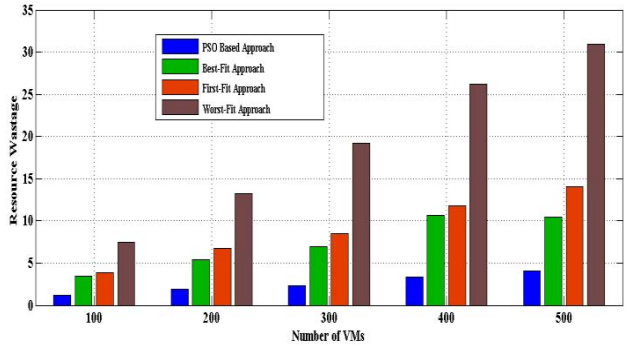

Figure 3: Total Resource Wastage v/s Number of VMs

Case II: Number of Active Servers v/s Number of VMs

Figure 4 shows the number of Servers Used v/s Number of VMs requests. The number of Servers Used is the total number of active servers to run all the VMs that host the cloud services. The results suggest that the PSO based approach always uses the least number of servers as compared to other approaches giving it an edge above others. As expected, the Worst-Fit algorithm always activates the maximum number of servers. The number of servers used in First-Fit and Best-Fit approach is nearly equal.

Case III: PSO over Generations

Figure 5 shows the convergence of the PSO algorithm. The program is run for 100 iterations. The number of VMs and servers is 100 where as the number of particles is 500. As can be seen PSO tries to minimize the resource wastage over iterations. Initially the convergence is very fast followed by a gradual move afterwards. Figure 6 and 7 shows the convergence of the swarm for two more such experiments. In Figure 6, the number of VMs and servers is 200, where in Figure 7, the number of VMs and servers is 300. The number of particles is 1000 and 1500 respectively. Similar to the case reported in Figure 5, here as well, the same pattern is observed with the PSO converging to a good solution. The convergence time varies from 25-65 iterations depending on size of the swarm and can be rated as fast.

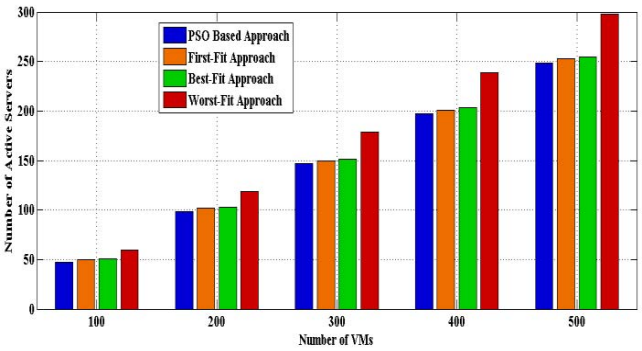

Figure 4: Number of Active Servers v/s Number of VMs

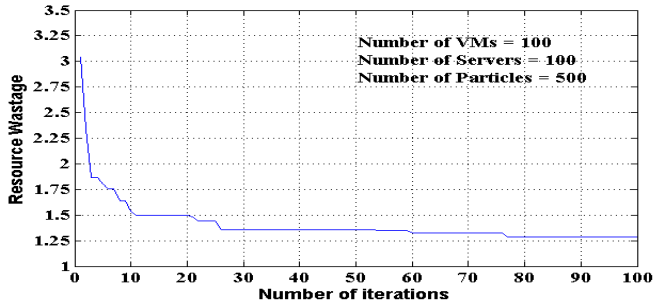

Figure 5: PSO Convergence over iterations

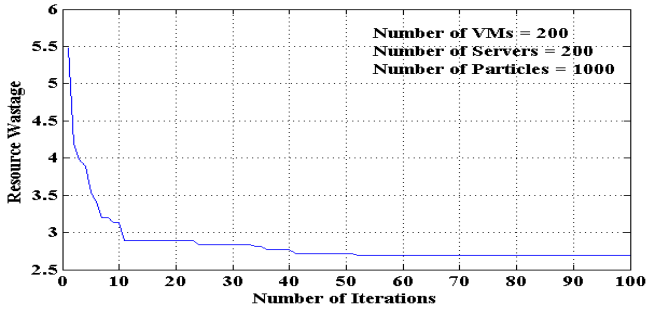

Figure 6: PSO Convergence over iterations

From the results obtained, it can be concluded that the proposed PSO based approach always performs better than BF, FF and WF strategy in all cases. The performance is measured both in terms of resource wastage and number of servers used. Further, from the experiments it has been observed that using PSO the resource wastage value decreases over generations and an optimum value is obtained owing to the reason that appropriate server is selected for placement of a particular VM resulting in lower resource wastage. The ability of the PSO to explore the vast solution space in a small time ensures these best results.

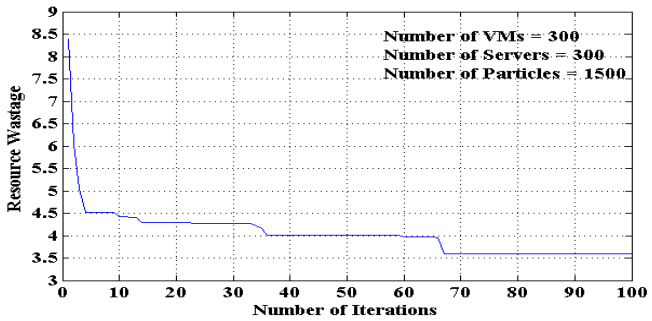

Figure 7: PSO Convergence over iterations

VII. CONCLUSION

Cloud computing is a new computing paradigm that offers a huge amount of storage and compute resources to the masses with relatively lower cost. With the increasing prevalence of large scale cloud computing environments, how to efficiently place VMs into available computing servers has become an essential research problem. This is important both from the point of view of minimizing the resource wastage as well as energy consumption as with the optimum number of servers used, substantial energy can be saved. The proposed work focuses on PSO based VM scheduling strategy in order to minimize the total resource wastage and servers used. Simulation experiments were conducted to observe the allocation of VMs to the servers and to evaluate the proposed algorithm with respect to scalability and performance. The results are compared with the Best-Fit, First-Fit and Worst-Fit placement strategies. The work can be further extended by inclusion of some other parameters for VM placement e.g. energy or availability with the possibility of using other swarm intelligence techniques explored. In addition a detailed comparative analysis can be undertaken with some current popular VM scheduling models.

REFERENCES

[1] M. A. Rappa, **"**The utility business model and the future of computing services.**",** IBM Systems Journal, vol. 43, no. 1, pp. 32-42, 2004.

[2] R. Buyya, V. Christian, and S. T. Selvi, *Mastering cloud computing***,** Tata McGraw-Hill Education, 2013.

[3] A. Wigderson**, "**P, NP and mathematics–a computational complexity perspective.**",** Proceedings of the 2006 International Congress of Mathematicians, 2006.

[4] J. Békési, G. Galambos and H. Kellerer, "A 5/4 linear time bin packing algorithm**",** Journal of Computer and System Sciences, vol. 60, no. 1, pp. 145-160, 2000.

[5] S. Srikantaiah, A. Kansal, and F. Zhao, **"**Energy aware consolidation for cloud computing**",** Proceedings of the 2008 conference on Power aware computing and systems, vol. 10, 2008.

[6] N. Bobroff, A. Kochut and K. Beaty, **"**Dynamic placement of virtual machines for managing sla violations", 10th IFIP/IEEE International Symposium on Integrated Network Management, 2007.

[7] D. M. Quan, et al., **"**Energy efficient resource allocation strategy for cloud data centres**"**, Computer and Information Sciences II, Springer London, pp. 133-141, 2012.

[8] K. Dhyani, S. Gualandi, and P. Cremonesi. **"**A constraint programming approach for the service consolidation problem.**",** Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Springer Berlin Heidelberg, pp. 97-101, 2010.

[9] S. Chaisiri, Bu-Sung Lee and D. Niyato, **"**Optimal virtual machine placement across multiple cloud providers**",** Services Computing Conference, APSCC 2009.

[10] Mi Haibo, et al., **"**Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers**",** Proceedings of IEEE International Conference on Services Computing (SCC), 2010.

[11] A. Beloglazov, J Abawajy and Rajkumar Buyya, **"**Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing**"**, Future Generation Computer Systems vol. 28, no. 5, pp. 755-768, 2012.

[12] Y. Gao et al., **"**A multi-objective ant colony system algorithm for virtual machine placement in cloud computing**",** Journal of Computer and System Sciences, vol. 79, no. 8, pp. 1230-1242, 2013.

[13] SS Patnaik and A .K. Panda, **"**Particle swarm optimization and bacterial foraging optimization techniques for optimal current harmonic mitigation by employing active power filter**",** Applied Computational Intelligence and Soft Computing, 2012.

[14] J. Kennedy, **"**The particle swarm: social adaptation of knowledge**"**, Proceedings of IEEE International Conference on Evolutionary Computation, 1997.

[15] K.E. Parsopoulos and M. N. Vrahatis, *Particle swarm optimization and intelligence: advances and applications,* Hershey, Information Science Reference, 2010.

[16] A. Rezaee Jordehi and J. Jasni, **"**Parameter selection in particle swarm optimization: a survey**"**, Journal of Experimental & Theoretical Artificial Intelligence, vol. 25, no. 4, pp. 527-542, 2013

[17] J. Kennedy and R. Eberhart, **"**Particle swarm optimization**"**, Proceedings of IEEE international conference on neural networks, vol. 4, no. 2, 1995.

[18] S Wang et al., **"**Particle Swarm Optimization for Energy-Aware Virtual Machine Placement Optimization in Virtualized Data Centers**"**, Proceedings of International Conference on Parallel and Distributed Systems (ICPADS), 2013.