International Journal of Research in Advent Technology, Vol.6, No.11, November 2018
E-ISSN: 2321-9637
Available online at www.ijrat.org

A Virtual Machine Hybrid Load Balancing Algorithm for Cloud Data Center Using Genetic Learning with Particle Swarm Optimization

Mala Yadav[1] and Jay Shankar Prasad[2]
School of Computer and Information Science
MVN University Palwal-121105
Haryana, India
E-mail: 12ca9002@mvn.edu.in[1]
E-mail: jayshankar.prasad@mvn.edu.in[2]

Abstract- Cloud computing brings a new era in internet computing. Cloud data centers plays a key role in cloud computing. Usage of Cloud data centers are highly dynamic and unpredictable. Unpredictability and dynamicity occurs due to user's irregular resource usage patterns, VM's fluctuating resource usages and user's arrivals and departure unstable rates. These issues generate unbalanced loads on cloud data centers, which lead to performance degradation and service level agreement violations. An effective load balancing mechanism must be required to address these challenges. Load balancing strategy ensure that dynamic workload is distribute across multiple nodes in equal amount at any instant of time, so that no single resource is either heavily loaded or underutilized. This paper proposes a hybrid load balancing strategy for cloud data center using Genetic Algorithm with Particle Swarm Optimization (GAPSO)Algorithm. Simulation of proposed load balancing strategy is performed using Cloud Analyst tool. Experimental result analysis of the algorithm is very inspiring. Considerably the results of the proposed algorithm are compared and outperformed the traditional algorithm like Round Robin(RR), Equally Spread Current Execution(ESCE) and soft computing approaches like Genetic Algorithm (GA).

Keyword- Cloud Computing; Load Balancing; Cloud Analyst; Genetic Algorithm; PSO Algorithm.

1. INTRODUCTION

National Institute of Standardized Technology (NIST) defines "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models" [1, 2]. In traditional data centers applications are tied to specific physical servers and leads to upper-bound workload. So traditional data centers have low resource utilization, less efficient, wastage of energy high and expensive to maintain. Cloud data centers [3,4] become more flexible, secure and provide better support for on-demand allocation. It hides server heterogeneity, enables server consolidation, and improves server utilization [5,6]. Virtualization technology revive the working strategy of traditional data centers. Virtualization helps in slicing a single data center or high power server to act as multiple machines. In virtualization technology, Cloud computing has been invented as an umbrella term to define a kind of refined on-demand computing services primarily presented by commercial providers, such as Amazon, Google, and Microsoft [7]. Despite glorious future of the cloud computing, many crucial problem needs to be addressed for the actual realization of the new technology.

The main goal of cloud computing is the efficient use of distributed resources to achieve a higher throughput, performance and solving large scale computing problems [8]. To achieve these goals, improving the general performance of system, maintain stability, availability and some other features for a cloud computing data center, an effective load balancing algorithms is required. Load balancing is one of the central and challenges issue in distributed systems like grid-based systems and cloud computing [9].

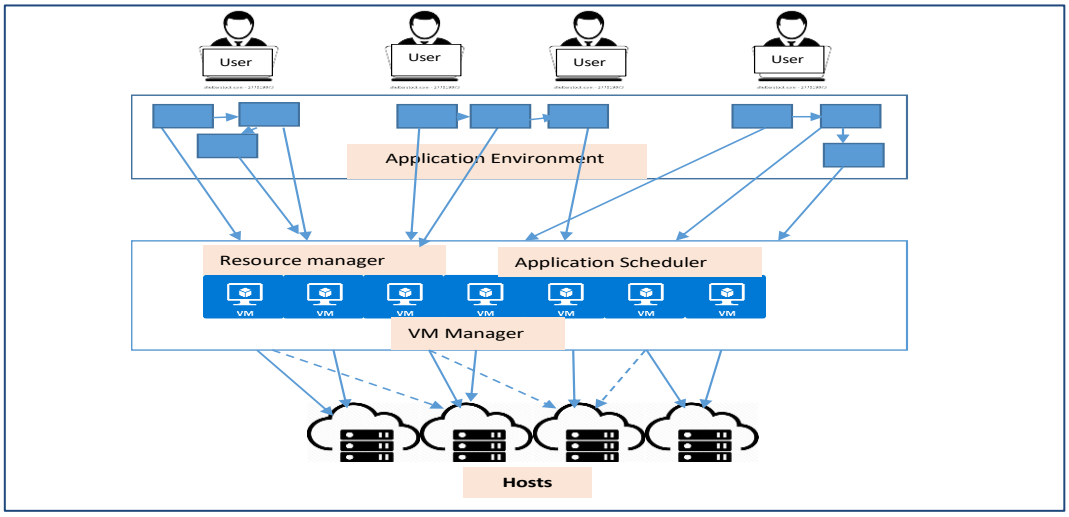

Fig. 1. User, Application, VM and Host relationship in Cloud Data Center

User, application, VM and host relationship in cloud data centers demonstrated in Fig. 1. The user applications are executed on VMs and may have dependencies between them. The hosts at the bottom have the physical resource for actual application execution like CPU, memory and storage resource. Above the hosts, the server virtualization platform available like XEN and VMware. These platforms make the physical resource virtualized and manages the VMs. Each host could be allocated with multiple VMs, and each VMs can handle multiple applications. Load balancing algorithms can be implemented at the application level and the VM level. At the application level, the load balancing algorithm integrated with Application Scheduler, and at the VM level, the load balancing algorithm integrated with VM Manager.

The main objective behind load balancing is to distribute the local workload evenly to the entire cloud to ensure at any instant moment of time no overloaded processor or resources is present in entire network. It is used by Cloud service provider (CSP) in its own cloud computing platform to provide a high efficient solution for the user. Thus load balancing helps to distribute any number of application requests, into any number of resources as service located in data centers through cloud service providers. For this reason, a good load balancing algorithm must be dynamic and adapt to the environment

In this paper a hybrid approach with Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) based on priority of available VM has been proposed. Combination of these two algorithms solves the load balancing problem by the help of dual optimization strategy. CloudAnalyst a CloudSim based visual modeler has been used for simulation and analysis of the proposed technique. Response time, processing time, throughput, power consumption, power density, and cost are considered as performance metrics to evaluate the obtained result with state-of–the-art methods. The experimental result remarkably optimizes the entire system load.

The rest of paper is planned in various sections. Section 2, about Related work. Section 3, Introduces the Cloud Analyst features. In Section 4, Genetic and PSO in the perspective of load balancing has been discussed. Section 5, Presents the analysis of results. Finally, Section 6 concludes this paper.

2. RELATED WORK

In NP-hard problem, finding the optimal solutions for algorithms is expensive. Hence, many of the proposed algorithms focus on searching approximate solutions for VM load balancing. As per these criteria algorithms are classified in three categories heuristic, meta-heuristic and hybrid.

Heuristic: Heuristic are set of constraints that are applied to find a good solution for a particular problem [10]. The constraints specified are always problem dependent and are planned as to get a solution in a limited time. Algorithms have various constraints, like number of migrations, SLAs, cost, energy etc. Optimization algorithm are created in diverse manners using these constraints. Heuristic algorithms are advantageous in many ways as they can find a satisfactory solution efficiently in limited time and cost, implementation is easier to meta-heuristic algorithms. Greedy algorithm is a type of heuristic algorithms and is applied in [11] [12] [13]

to quickly obtain a solution for online scheduling scenario.

Meta-heuristic: Meta-heuristic algorithms are generally designed for a general purpose problem [10]. So, meta-heuristic algorithms have a set of uniform procedures to design and solve problems. Meta-heuristic algorithms like Genetic algorithms, Ant Colony Optimization, Particle Swarm Optimization and Honeybee Foraging Algorithms are inspired from nature. These algorithms are based on population evolutions and obtaining the best population in each evolution and keep it into next evolution. Ant Colony Optimization and Particle Swarm Optimization are combined in [14] to deal with VM load balancing. Proposed strategies results show that better load balancing achieved as compared to heuristic algorithms. As metaheuristic algorithms have large solution space so require more time to find the final solution. However, meta-heuristic processes are usually stochastic and their convergence time and solution results depend on the nature of problem, initial configurations and the way to search the solutions. Meta-heuristics achieve better results than traditional heuristics.

Hybrid: Hybridization of algorithm can be performed in various manners to find the optimized solution such as heuristic algorithm is applied for initial VM placement and after that meta-heuristic algorithm applied in migration to optimize the placement of VMs. Instead of this meta-heuristic algorithms applied at initial stage to generate a set of solutions, and after that heuristic algorithms applied to find the optimized solution based on previous solutions. Hybridization in either way, reduces time, cost and solution space. However, the complexity of implementation increases. Thiruvenkadam et al. [15] proposed a hybrid genetic algorithm using the first approach.

Literature Armstrong et. al. [16] explain various approaches of Load balancing which uses Minimum Execution Time(MET) to assign order to each job in arbitrary manner to the nodes for efficient execution, irrespective of considering the existing load on that node. Data centers are using some of existing scheduling techniques like Min-Min, Round Robin and FCFS for load balancing. Yang Xu et. al. [17] proposed an intelligent method for load balancing. Proposed novel model improve cloud computing performance by balancing data distribution in data-intensive applications, like distributed data mining.

Comparative study indicates linear algorithm execution is efficient but organization of large number of cloudlets is inefficient and unmanageable whereas genetic algorithm and swarm intelligence algorithms find the best optimal solution of the problem by mapping of parameters.

3. CLOUD ANALYST

CloudAnalyst [18] is the open source simulation developed on CloudSim[19]. CloudAnalyst provides an efficient and reliable simulation environment so that user can focus on the main simulation process. CloudAnalyst enables users to repeatedly execute simulations experiments with slight parameters variations in a quick and easy manner. The main features of CloudAnalyst are: Graphical User Interface (GUI), Simulation with a high degree of configurability and flexibility, Repeatability of experiments, Graphical output. Use of consolidated technology and ease of extension.

The design of CloudAnalyst is shown in Fig. 2. CloudAnalyst is built on top of CloudSim toolkit, with functionalities of Internet Application.

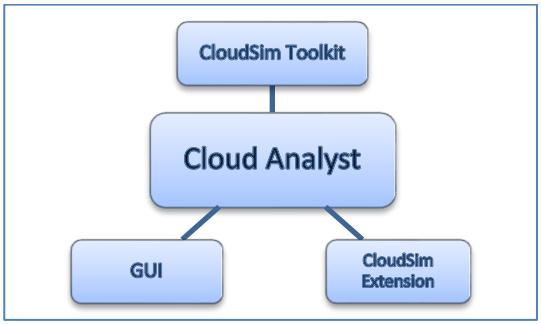

Fig. 2. CloudAnalyst Design

The main components are:

GUI Package. It acts as the front end controller for the application, managing screen transitions and other user interface activities.

Simulation. It holds the simulation parameters for creating and executing the simulation.

UserBase. It models a user base and generates traffic representing the users.

DataCenterController. It controls the data center activities.

Internet. It models the Internet and implements the traffic routing behavior.

InternetCharacteristics. It maintains the characteristics of the Internet during the simulation, including the latencies and available bandwidths between regions, the current traffic levels, and current performance level information for the data centers.

VmLoadBalancer. It models the load balance policy used by data centers when serving allocation requests. Default load balancing policy uses a round robin algorithm, throttled load balancing policy etc.

CloudAppServiceBroker. It models the service brokers that handle traffic routing between user bases and data centers. The default traffic routing policy is routing traffic to the closest data center in terms of

network latency from the source user base. In addition, an experimental brokerage policy for peak load sharing is implemented on CloudAnalyst.

4. LOAD BALANCING USING GENETIC ALGORITHM WITH PARTICLE SWARM OPTIMIZATION SCHEDULING

Load Balancing process is to be done for effective resource utilization so that the response time of the job to be improved. While developing strategy for load balancing the main points to be considered are evaluation of load, stability of different system, performance of system, node interaction, type of work to be transferred, selecting of nodes [20]. Load may be in terms of CPU load, amount of memory used, delay or Network load. This can be considered as an optimization problem where loads are distributed among the available servers to achieve an effective throughput. Soft computing approach considered as a effective optimization tool. GA (Genetic Algorithm) and PSO (Particle Swarm Optimization) plays an important role in the field of bio-inspired optimization

In load balancing mechanisms consideration of task length, priority and inherent capabilities of the resources plays an important role to avoid overloading of machine and late response. The proposed algorithm considers priority of task and problem of late response. Objective is to minimize the overloads as well as task migrations. Genetic algorithms used for load balancing [21] utilizes the resources efficiently. Hence, to reduce the response time it requires to tune the genetic operators and particle swarm optimization. GA and PSO are hybridized in a cascade manner [22]. The main loop of the algorithm is composed of two cascading layers, the first for generating particle by applying GA and the second for positioning particle at suitable VM as per a normal PSO algorithm. GA and PSO are hybridized in a way, which establishes a positive feedback loop to accelerate the population to locate the optimum.

4.1 *Genetic algorithm learning for particle generation*

In initial phase problem has been formulated and on available VM genetic algorithm [23] has been used. GA is basically inspired by the nature and select the most suitable strings from the organized information using crossover and mutation under the constraints of fitness function.

Three basic operators Selection, Crossover and Mutation of GA work intelligently and provide higher quality chromosomes(particles). In GA two chromosomes are randomly picked to exchange their component genes through crossover, and some genes in the chromosomes are randomly varied through mutation.

1. Selection: Chromosomes are selected randomly from initial population. Fixed bit string used by GA to represent individual solution. That's why, all the possible solutions in the solution space are encrypted into binary strings.
2. Crossover: Chromosomes undergoes a random single point crossover and generates a new pair of individuals. Fitness function calculate the fitness value of each individual chromosome. The objective of this step is to select best fitted pair of individuals for crossover.
3. Mutation: Bits of the chromosomes, are toggled from 1 to 0 or 0 to 1 according to the mutation value. Generally, a very small value (0.05) is selected as mutation probability.

4.2 *Positioning of particle by applying particle swarm optimization approach*

Particle swarm optimization (PSO), a nature-inspired optimization technique, has attracted significant attention since introduced by Kennedy and Eberhard [24] [25] in 1995. Simulating the social behavior of birds flocking or fish schooling, a population of particles in a PSO algorithm cooperates and interacts to search for solutions in the problem space. Owing to its conceptual simplicity and high efficiency, PSO has been successful in solving a variety of problems in many areas. PSO [26] is based on the principle that each solution can be represented as a particle in the swarm. Each particle has a position in the search space, which is represented by a vector. Particles move in the search space to search for the optimal solutions. Therefore, each particle has a velocity. During the movement, each particle updates its position and velocity according to its own experience and that of its neighbors. The best previous position of the particle is recorded as the personal best pbest, and the best position obtained by the population thus far is called gbest. Based on pbest and gbest, PSO searches for the optimal solutions by updating the velocity and the position of each particle. The algorithm stops when a predefined criterion is met, which could be a good fitness value or a predefined maximum number of iterations. Fitness function depends on goal in optimization problem. The proposed algorithm will obtain the global optimized solution of load balancing.

4.3 *Proposed Algorithm*

1: First of all, Initialize Vm and Current Allocation Count list

2: Get Vm list from Datacenter controller

3: Request is allocated on available Virtual Machine

$$Vm\ state \leftarrow\ vmStatesList.get(Vm\_ID);$$

5: If VM is available then Genetic processing and go to step 6

6: Initialize a cloudlet of processing unit and encode them into binary strings and generate packets as per fitness.

7: Fitness: Evaluate the fitness value of each task using

$$MaxLim = \ (2^{maxp} - 1)$$
$$Select\ RandomVal = [0,1] * MaxLim.$$

Select the fittest Vm from each patch on the basis of condition i.e. mean fitness value of machine. At each algorithm iteration, the fittest Vm will be chosen to assign tasks in Vm.

8: $while\ (no\ of\ iterations = maxiterations)$or optimum solution is found Do:

Selection: Consider chromosome with lowest fitness.

Calculate fitness value for each particle

$If\ (fitness\ >\ Maximum\ Power)$ then Solution find else increase number of process and start iteration again.

Crossover: Single point crossover by randomly selecting the crossover point for new offspring.

Mutation: Mutate new off spring with new index value and a mutation probability.

9: Compute tasks positions according to velocity (Particle Swarm Optimization) with parameters fitness value, threshold and number of success process from genetic learning.

10: Choose the particle with the best fitness value of all the particles as the gbest

For each particle

10(a): Calculate particle velocity v[ ]:

$$v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[])$$

$$vValue = \frac{(vmax * aParticle.getConflicts(\ ))}{worstResults};$$

10(b): Update particle position:

$$present[] = persent[] + v[]$$

$$int\ changes = (int)Math.floor(Math.abs(destination.getVelocity()));$$

11: Define destination for tasks with index id and make processing as a board using method:

(i) clearBoard(board);

(ii) plotQueens(board);

12: Maintain results log file

5. SIMULATION RESULT AND ANALYSIS

Simulation performed under various configuration of cloud analyst. A hypothetical configuration has been generated using CloudAnalyst. Where, the world is divided into 6 ‘’Regions” that concur with the 6 main continents in the World. Six “User bases” modeling a group of users representing the six major continents of the world is considered.

The applied load balancing policy is being executed by using the Optimize Response Time service broker policy. The simulation is being conducted with 10 user base grouping factor in user bases and 10 request grouping factor in data centers. For analyzing the scenarios and results of the intended technique configuration of various components of cloud analyst tool need to be set. Round Robin (RR), Equally Spread Current Execution Load (ESCE), Genetic Algorithm (GA) and Proposed Genetic Algorithm and Particle Swarm Optimization(GAPSO) approach for load balancing has been implemented. Various statistical metrics derived as the output of the simulation.

Table 1: Simulation Configuration and calculated overall average response time in (ms)

S.No.	Simulation Configurations	DC Specification	Average Response Time in (ms)			
			RR	ESCE	GA	PGAPSO
1	SC1	Userbase-31,DC-5, VM-5	384.85	380.16	379.72	263.40
2	SC2	Userbase-42,DC-5, VM-5	337.83	340.68	344.14	234.74
3	SC3	Userbase-45,DC-6, VM-5	345.81	344.90	353.88	234.36

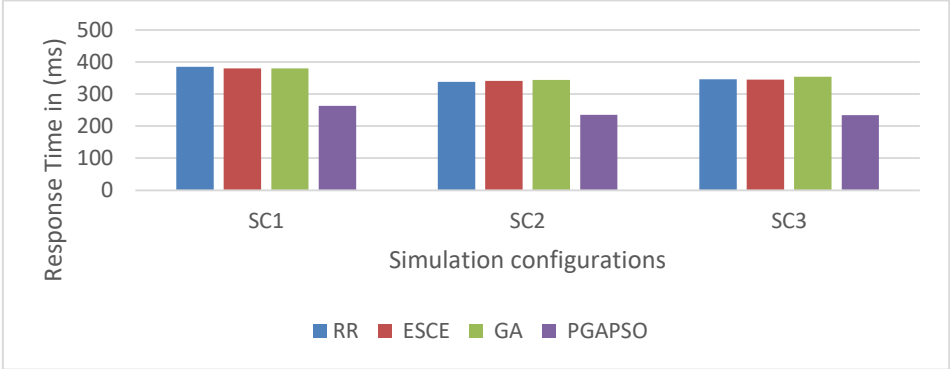

Fig. 3: Response Time Analysis under different simulation configurations

Table 2: Simulation Configuration and calculated overall average Processing time in (ms)

S.No.	Simulation Configurations	DC Specification	Average Processing Time in (ms)			
			RR	ESCE	GA	PGAPSO
1	SC1	Userbase-31,DC-5, VM-5	272.95	267.99	267.29	153.55
2	SC2	Userbase-42,DC-5, VM-5	231.26	234.61	237.87	128.00
3	SC3	Userbase-45,DC-6, VM-5	243.32	243.07	252.43	132.40

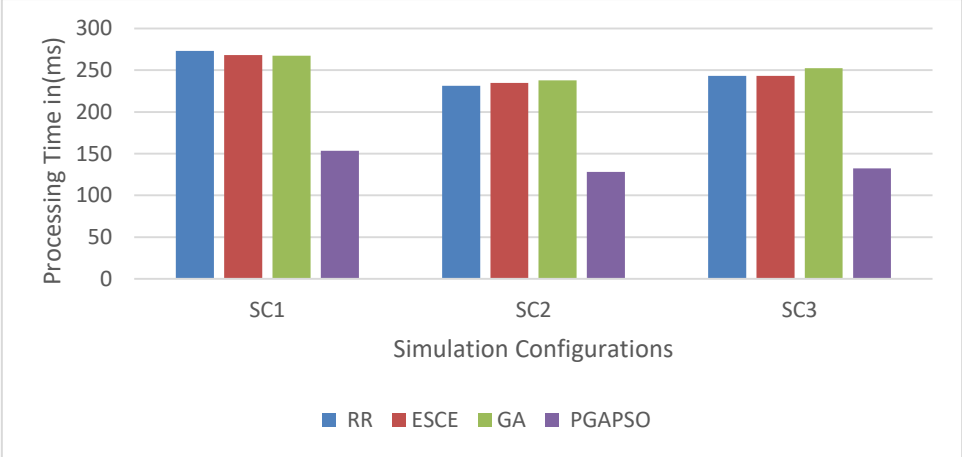

Fig. 4: Processing time of the algorithms under different simulation configurations

Fig. 3 analysis depicts that the Proposed GAPSO response almost 35% faster than the RR, ESCE and GA load balancing algorithms in different configurations using different userbases. The processing time in milliseconds(ms) depicts the time taken by a particular algorithm to balance the load. The processing time analysis under varying user base of the data center is shown in Fig. 4 and shows Proposed GAPSO almost 30 % faster. The service request time measures the data center performance. The service request time taken through proposed algorithm is very less as compared to other methods.

The dynamic scheduling of Proposed GAPSO allocates VMs such that it achieves better load balancing. It does load balancing such that VM migrations are reduced since it first checks the availability of the VMs then generate packets of the available tasks using genetic learning and decide the allocation of the tasks to the appropriate VM using particle swarm optimization. Proposed GAPSO performs the same tasks in less number of VMs.

Fig. 5 depicts Throughput analysis that Proposed GAPSO outperforms and show the overall computational effectiveness and efficiencies.

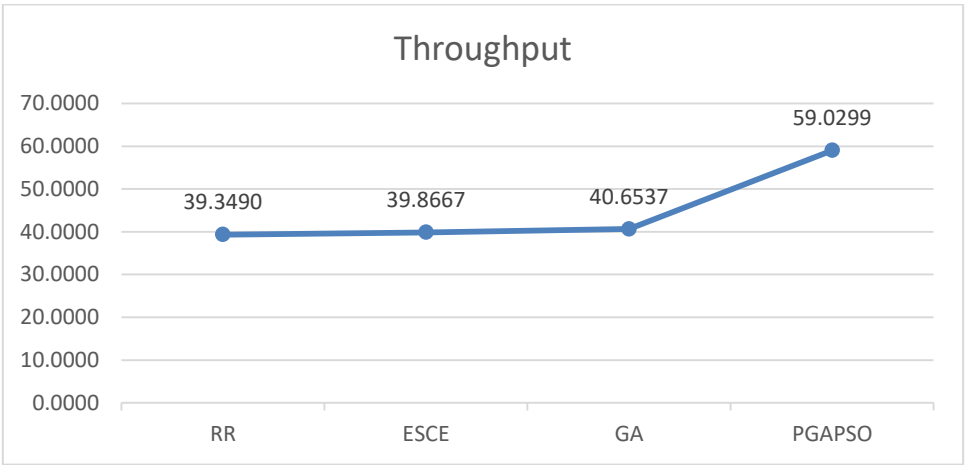

Fig. 5: Throughput Analysis comparison of algorithms

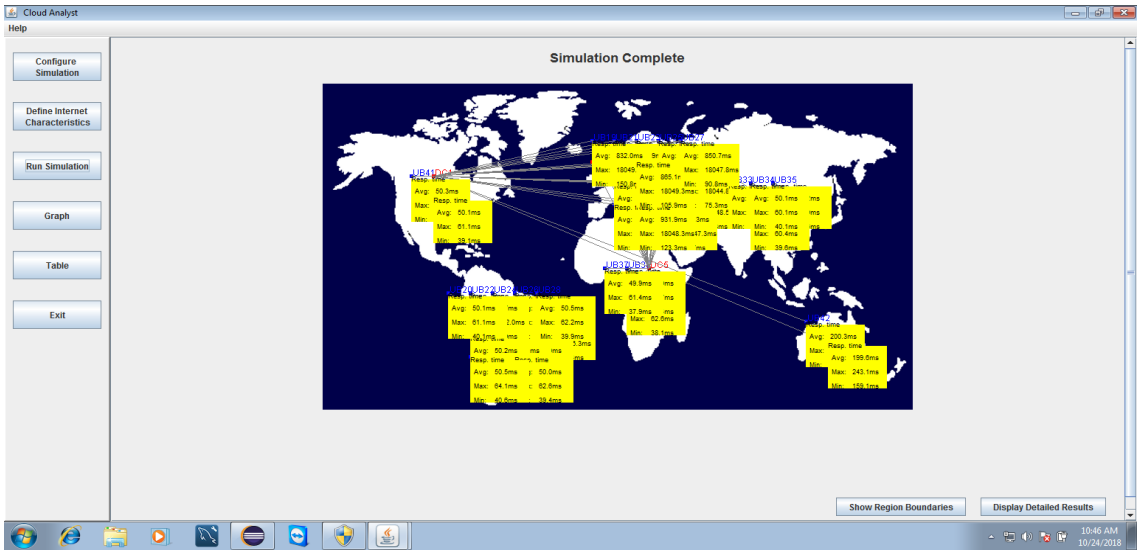

Fig. 6: Simulation Window Snapshot of Cloud Analyst

Fig. 6 depicts the simulation and task assigning at nodes. On other hand it is also found that there is no extra cost issue in proposed algorithm total virtual machine cost and data center cost is same for all the approaches.

Energy consumption is also a serious issue for data centers. Energy consumption of resources is not proportional to their utilization. Simulation results validate the effectiveness of the proposed load balancing in minimizing the total energy consumption and carbon emission of data center and plays a role in green computing by saving energy.

Fig. 7 shows that Power Consumption in Proposed GAPSO is low. Power Density specifies the amount of power consumed by the request size from the overall power consumption.

Table 3: Simulation Configuration and calculated overall average power density in (kw)

			Power Density			
S.No.	Simulation Configurations	DC Specification	RR	ESCE	GA	PGAPSO
1	SC1	Userbase-31,DC-5, VM-5	1.4639E-06	1.4639E-06	1.4639E-06	1.1194E-06
2	SC2	Userbase-42,DC-5, VM-5	1.9833E-06	1.9833E-06	1.9833E-06	1.9833E-06
3	SC3	Userbase-45,DC-6, VM-5	2.7500E-06	2.7500E-06	2.7500E-06	2.6250E-06

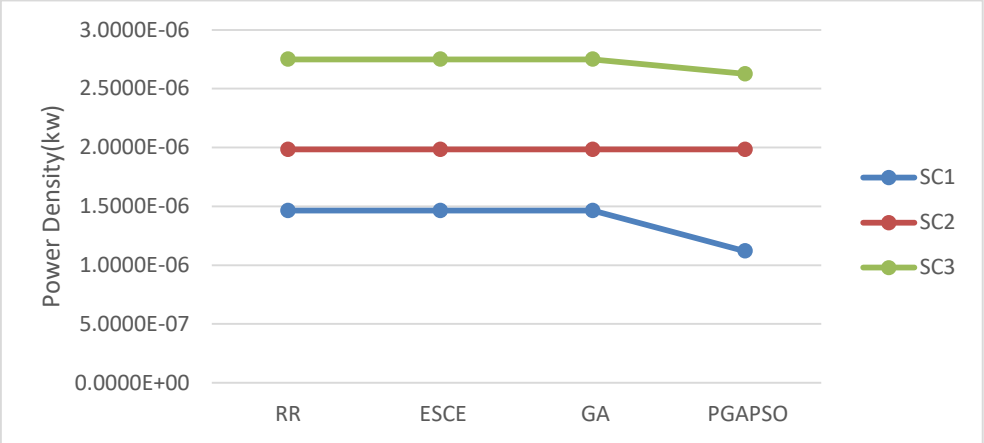

Fig 7: Consumption of Power Density in(kw) under various analysis

The several scenarios in this paper are considered for experimentation and compare the result of existing

Round Robin(RR), Equal Spread Current Execution (ESCE), Genetic Algorithm (GA) and Proposed Genetic Algorithm Particle Swarm Optimization (GAPSO) This proposed technique will be adapted in the cloud computing environment for efficient and better scheduling to cloud resources.

6. CONCLUSION

This paper proposes a hybrid approach of Genetic Algorithm (GA) and Particle Swarm Optimization(PSO) Algorithm based load balancing strategy for VMs of Datacenter. GAPSO hybrid meta-heuristic paradigm delivers efficient scheduling of tasks on existing resources, so that completion time of tasks become minimized and saves energy. Effective packing of tasks maximizes the resources usage and minimize VM migration. Result Analysis demonstration indicates that the proposed strategy for load balancing outperforms a few existing techniques like Round Robin(RR), Equally Spread Current Execution(ESCE) and soft computing approaches like Genetic Algorithm (GA). The results given by the proposed work is much better than the other methods used previously. Performance analysis performed only by using limited number of jobs and resources. So, future work could go in the direction to analyze the performance of the proposed algorithm in real environment and variations of crossover, mutation, and selection can generate high-quality offspring and selection strategies could be applied with other soft computing approaches for getting more efficient results.

REFERENCES

[1] http://csrc.nist.gov/publications/nistpubs/800-146/sp800146.pdf.

[2] Puthal, D., et al., *Cloud Computing Features, Issues, and Challenges: A Big Picture*, in *International Conference on Computational Intelligence and Networks (CINE)*2015, IEEE. p. 116-123.

[3] Trestian, R., Katrinis, K., Gabriel-Miro M., (2017). OFLoad: An OpenFlow-based Dynamic Load Balancing Strategy for Datacenter Networks, IEEE Transactions on Network and Service Management, vol.14, Issue 4, pp 192-807

[4] Tsygankov, M. and Chen, C. (2017). Network aware VM load balancing in cloud data centers using SDN, *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, Osaka, pp. 1-6., doi: 10.1109/LANMAN.2017.7972179

[5] Daniels, J. (2009). Server virtualization architecture and implementation. Crossroads, 16(1):8–12.

[6] Speitkamp, B., Bichler, M. (2010). A mathematical programming approach for server consolidation problems in virtualized data centers. IEEE Transactions on services computing, 3(4):266–278.

[7] Buyya, R., Broberg, J. and Goscinski A. (2011) CLOUD COMPUTING Principles and Paradigms, Jhon Wiley & Sons.

[8] Jadeja, Y. and K. Modi. *Cloud computing-concepts, architecture and challenges*. in *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*. 2012. IEEE.

[9] Rimal, B.P., E. Choi, and I. Lumb. *A taxonomy and survey of cloud computing systems*. in *INC, IMS and IDC, 2009.NCM'09. Fifth International Joint Conference on*. 2009.Ieee.

[10] Talbi, E.G. Metaheuristics: from design to implementation, vol. 74. John Wiley & Sons, 2009.
[11] Song, X., Ma ,Y., Teng, D. A load balancing scheme using federate migration based on virtual machines for cloud simulations. Mathematical Problems in Engineering 2015; 2015.
[12] Ni, J., Huang, Y., Luan, Z., Zhang, J., Qian, D. Virtual machine mapping policy based on load balancing in private cloud environment. Cloud and Service Computing (CSC), 2011 International Conference on, IEEE, 2011; 292–295.
[13] Tian W, Zhao Y, Zhong Y, Xu M, Jing C. A dynamic and integrated load-balancing scheduling algorithm for cloud datacenters. 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, IEEE, 2011; 311–315.
[14] Cho, K.M., Tsai, P.W., Tsai, C.W., Yang, C.S. A hybrid meta-heuristic algorithm for vm scheduling with load balancing in cloud computing. Neural Computing and Applications 2015; 26(6):1297–1309.
[15] Rouzaud-Cornabas J. A distributed and collaborative dynamic load balancer for virtual machine. European Conference on Parallel Processing, Springer, 2010; 641–648.
[16] Armstrong, T.R. and Hensgen, D. (1998). The relative performance of various mapping algorithms in independent runtime predictions, in proc. Of 7th IEEE Heterogeneous computing workshop (HCW 98). Pp. 79-87.
[17] Xu, Y., Wu, L., Guo, L., Yang, Z.C. and Shi, Z. (2008). An Intelligent Load Balancing Algorithm Towards Efficient Cloud Computing, in Proc. of AI for Data Center Management and Cloud Computing: Papers, from the 2011 AAAI Workshop (WS-11-08), pp.27–32.
[18] Wickremasinghe, B., Calheiros, R.N., and Buyya, R. (2010). Cloudanalyst: A cloudsim-based visual modeller for analyzing cloud computing environments and applications", inProc. of Proceedings of the 24th International Conference on Advanced Information Networking and Applications(AINA2010), Perth, Australia, pp.446-452.
[19] Calheiros, R.N., Ranjan, R., Beloglazov, A., Rose, C. and Buyya, R. (2011). Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, in Software: Practice and Experience(SPE), Volume 41, Number 1, ISSN: 0038-0644, Wiley Press, New York, USA., pp. 23-50.
[20] Alakeel, A. M., A guide to dynamic load balancing in distributed computer systems, in: IJCSNS International Journal of ComputerScience and Network Security, VOL.10 No.6., 2010, pp. 153–160.
[21] Goldberg, D.E. (1989). Genetic algorithms in search, optimization, and machine learning, Addison-Wesley.
[22] Gong, Y.J., Li, J.J., Zhou, Y., Li, Y., Chung, H.S.H., Shi, Y.H. and Zhang, J. (2016). Genetic learning particle swarm optimization, IEEE transactions on cybernetics 46 (10), 2277-2290.
[23] Dasgupta, K., Mandal, B., Dutta, P., Mandal, J.K. and Dame, S. (2013). *A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing*, Journal of ScienceDirect, Vol. 10, pp. 340 – 347.
[24] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization, in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. Perth, WA, Australia, pp. 1942–1948.
[25] Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory, in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, Nagoya, Japan, pp. 39–43.
[26] Xue, B., Zhang, M. and Browne, W.N. (2013). Particle swarm optimization for feature selection in classification: A multi-objective approach, IEEE *Trans. Cybern.*, vol. 43, no. 6, pp. 1656–1671.