2020 International Conference on Contemporary Computing and Applications (IC3A)
Dr. A.P.J. Abdul Kalam Technical University, Lucknow. Feb 05-07, 2020

Load Balancing in Cloud Computing using Mutation Based Particle Swarm Optimization

Ronak Agarwal
Computer Science & Engineering
GLA University, Mathura, India
ronakagrawal0504@gmail.com

Neeraj Baghel
Center for Advanced Studies
AKTU, Lucknow, India
nbaghel777@gmail.com

Mohd. Aamir Khan
Indian Institute of Technology
IIT Kanpur, India

2020 International Conference on Contemporary Computing and Applications (IC3A) 978-1-7281-5432-9/20/$31.00 ©2020 IEEE 10.1109/IC3A48958.2020.233295

***Abstract*— Cloud computing has emerged as a technology that grease tasks by the dynamic allocation of virtual machines. Users pay for resources based on their demand. A cloud provider has to face many challenges. One out of the essential problem is load balancing, which suffers from many issues like premature convergence, reduced convergence speed, at first chosen random solutions, and stuck in native optima. The proposed method considered the MakeSpan parameters to handle the problem related to existing met heuristic techniques. The proposed method focuses on the mutation-based Particle Swarm algorithm to balance load among the data centers. Here an efficient load balancing algorithm is developed to minimize performance parameters like MakeSpan time and improve the fitness function in cloud computing.**

Keywords—cloud computing, load balancing, particle swarm optimization, virtual machine.

I. INTRODUCTION

Cloud computing has risen as jargon in the area of high-performance computing and distributed environment. It provides users with on-demand services to a shared pool of resources over the Internet which is dynamically scalable. Cloud computing is being the most magnifying field in web technologies. It dynamically allocates the task on the virtual machine which is easily accessible to the user. One of the main challenges in cloud computing is load balancing. Load balancing is a term defined for sharing the load among the multiprocessors. The workload can be classified into various categories like CPU load, network load, memory capacity issue, etc. Meanwhile, in cloud computing, load balancing mechanisms are used to share the amount among virtual machines with the help of a virtual machine manager (VMM). Cloud information centers are usually designed to handle hundreds of loads that may lead to low resource utilization and wastage of energy therefore it needs to be optimized. Load balancing techniques like particle swan optimization (PSO) [1] are required for distribution of the workload among cloud centers to forestall a state of some nodes being overloaded, underloaded or may be idle. The workloads are mapped with various resources below the constraint of energy improvement [2]. Load balancing is effectively used for getting high and sufficient quality of services. The load balancer distributes client requests across multiple servers. They use the concept of virtualization with the help of hypervisors on a virtual machine manager (VMM).

In [3] contributed a systematized review of the different load balancing algorithms. The drawback of this research is that it did not talk about scheduling and load balancing techniques in Hadoop Map Reduce, which is an area of concern nowadays. In [4] went through suitable load balancing algorithms. They gave the fact that the new techniques should be more focused on energy saving and consumption. In [5] experimented with the load balancing techniques according to different parameters. In [6] a journal intelligent cloud algorithm is proposed for load balancing issues, including Ant Colony Optimization (ACO), Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC). They introduced a new concept of Ant Lion Optimizer (ALO) based mostly on cloud computing surroundings as an economical rule. In [7] FIFO works on a line of jobs. It is a default scheduler in Hadoop. The submitted professions are consecutively executed under a homogeneous group that accepts in Hadoop by default FIFO work scheduler.

Capacity scheduler. This standard was created by Yahoo in request to have a decent distribution of assets among the vast populace of bunch clients (Zaharia et al., 2010) [8]. For this reason, it uses lines with a partner framework of errand spaces. Vacant assets are designated to lines in accordance with the needs. On the off chance that there are vacant assets in certain lines, they're allocated to elective lines. Among a line, the need for employment is resolved to support the obligation point in time, occupation's classification, and need settings for the client's Service Level Agreement (SLA). When an opening inside the TaskTracker turns out to be free, the scheduler picks the activity with the longest holding up time from a line with an absolute bottom burden. Hereafter, the capacity PC equipment reinforces bunch sharing among clients, rather than among employments, similar to that of the case inside the honest PC.

Delay scheduler is a renewed scheduler that works on the locality problem. In the event that the delayed time turns out to be longer, to keep away from starvation, no neighbourhood task is permitted to plan. Longest Approximate Time to End (LATE)[9]. This scheduler was developed in response to boost the task latency for Hadoop in different surroundings. Speculative tasks are tasks that may progress in a very slower manner due to computer hardware race condition, high load, temporary retardation and the background processes or other slow processes. A slow task has identified this hardware and executes an equivalent backup task on a totally different node and this execution is known as speculative execution. If the copy of the task completes in less time as compared to the previous one, the overall job performance can improve. The LATE computer hardware assigns priorities to unsuccessful or

slow tasks to selects the quickest nodes for that speculative execution.

The deadline constraint scheduler [10] was planned so as to fulfil the client imperatives. The objectives of the schedule are to provide the option to the clients on whether the activity can be finished inside the given timeframe or not and continue with the execution of the given time frame and also to amplify the number of employments. In [11] the author gave a load balancing technique for energy consumption in cloud computing by analysing the life of a flock of birds, which generally call as 'cuckoos.' They utilized the Cuckoo Optimization Calculation. For choosing VMs from under-used and overused hosts the Minimum Migration Time (MMT) strategy is utilized. The Simulation results determined the yield by diminished vitality utilization. In [2] a genetic algorithm (GA) is proposed for load-balancing. The algorithm creates a "population" of possible problem sets and allows them to "evolve" over multiple generations to fittest solutions. In [12] gave a calculation named as Ant Colony Optimization for adjusting the heap ideally. This calculation takes motivation from the Ant states which works in a searching behaviour. A honeybee-based technique is given in [13] that minimize the waiting time in the queue by considering the tasks. Whenever a task is submitted to a VM, it updates the priority task and asks others tasks for the help in choosing an underloaded virtual machine. This algorithm minimizes the execution time and reduced the waiting time of tasks in the queue.

The main contribution of the proposed work is to resolve the issue of load balancing in cloud computing by decreasing the makespan time by using the mutation-based particle swan optimization algorithm. The result is validated as the makespan of the proposed MPSO is lesser than the existing PSO. The fitness function is also improved. The dummy jobs are removed.

This paper has 3 more sections as follows: Section 2 describes the proposed technique. Section 3 describes the simulation and results. Section 4 describes the conclusion and future work of the proposed method.

II. PROPOSED TECHNIQUE

In this section, the Mutation Based PSO (MPSO) is described. The proposed method has a purpose to accentuate the excavate capability of the search process. Generally, high-end servers contain a vast amount of computing power and storing resources. These servers usually communicate with one another with the help of a high bandwidth intercommunication network. Therefore transmission delay does not play much significant role in cloud computing. In this environment, each user utilizes cloudlets with help of the Internet. The cloud service provider has to take care of allocating and de-allocating the resources to high-end clients. The job of the user is to propagate between several data centers (DCs) in the cloud. Each DCs divides user jobs into various jobs and allocates them between available processing elements in their (DCs). The proposed load balancing technique is responsible for assigning of jobs into available (DCs) efficiently with an aim to reduce the MakeSpan time and improve fitness function.

A. Mutation based Particle Swarm Optimization

The proposed method used mutation-based particle swarm optimization to reduce the makespan time and to improve fitness function to optimize load balancing in cloud computing. The flowchart of the proposed MPSO algorithm is shown in Fig.1

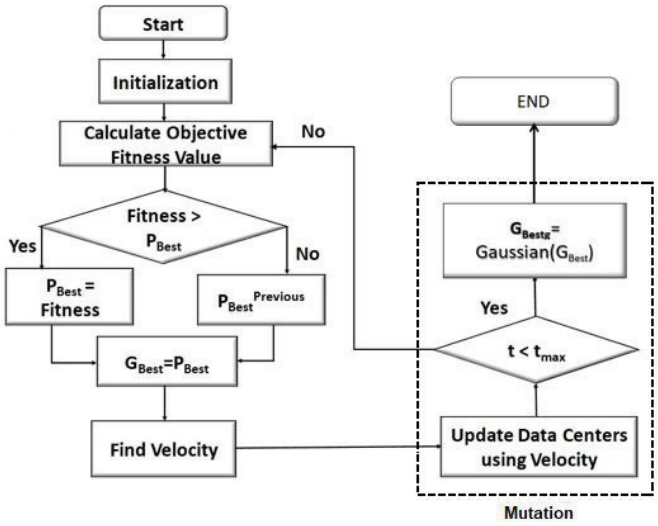

Fig. 1. The Flowchart of the proposed MPSO algorithm

Figure 1 illustrates the flowchart of the proposed MPSO algorithm where the first initialization of each particle takes place and each particle has a random velocity. Then fitness value is calculated for each particle using a fitness function. After that, velocity calculation takes place using eq1.

$$v = v + c1.rand().(P_{Best} - Present) + c2 * Random().(G_{Best} - Present) \quad (1)$$

Then the particle's position is updated using eq2.

$$Present[] = Present + v[] \quad (2)$$

The mutation is now applied to the best global value and the new best global value is calculated using eq3.

$$Gbest_{g1}(d) = Gbest_g(d) + (X_{max}(d) - X_{min}(d)) * Gaussian(o.h) \quad (3)$$

When the maximum value of iteration is matched then the process gets terminated. Where v vector is termed as the particle velocity vector, Present[] is the current solution. P_{best} is the particle's best solution and G_{best} is the best global value, rand () is a random function for generating values. The new fitness value is then compared with its best personal value and best global value. The fitness value then takes a value of the best P_{best} value or G_{best} value. In the case of a similar value, the fitness function takes the existing P_{best} value.

B. Applying Mutation for Optimized Result

The value of the whole population improves if the value of G_{best} improves. So mutation will perturb the best global value randomly. Also, the quality of global best values can improve as a result of perturbation and improved value is accepted. Stronger mutation during the beginning of an optimization

supports smaller displacements towards the end aid in fine-tuning extreme values and sampling of the search space. The mutation will remove the dummy jobs i.e. it will remove those solutions which are taking more response time and do not fit into the particles. These kinds of jobs are then removed. Thus we have a set of fittest global values. Poorly selected particles are removed from the tasks workspace now.

The Gaussian mutation operator is applied to the G_{best} value to improve the search space as it is very flexible and tunes the extreme values.

The proposed model will return the last outcome when the characterized number of cycles met or the wellness stays steady for over ten emphases.

III. SIMULATION AND RESULT

A. Implementation Details

This mutation-based PSO algorithm (MPSO) is implemented on an i7 processor with an 8GB RAM working on a 64-bit operating system using CloudSim 3.0.3 Tool. CloudSim is configured with the help of eclipse-mars. This tool is effective for developers to judge cloud application demand workloads and server's geographical distribution. CloudSim may run on Linux and Windows systems. This experiment minimizes the MakeSpan (MS) and improves the fitness function.

B. Parameters setting

Different Parameters are taken here and their comparison is done as shown below. The value of the MakeSpan of MPSO is compared with existing PSO for 10 data centers DC_s as shown in Fig2, 15 DC_s shown in Fig.3 and 20 DC_s shown in Fig.4. The number of cloudlets is taken as 50, 100, 150 and 200 for each number of DC_s.

Fig. 2. PSO and MPSO makespan time for 10 Data Centers

Figure 2 shows the value of makespan is calculated for 10 data centers with each 50, 100, 150 and 200 cloudlets for both MPSO and PSO approaches.

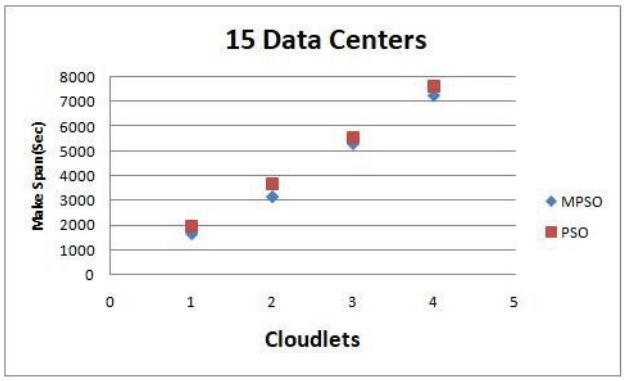

Fig. 3. PSO and MPSO makespan time for 15 Data Centers

Figure 3 shows the value of makespan is calculated for 15 data centers with each 50, 100, 150 and 200 cloudlets for both MPSO and PSO approaches.

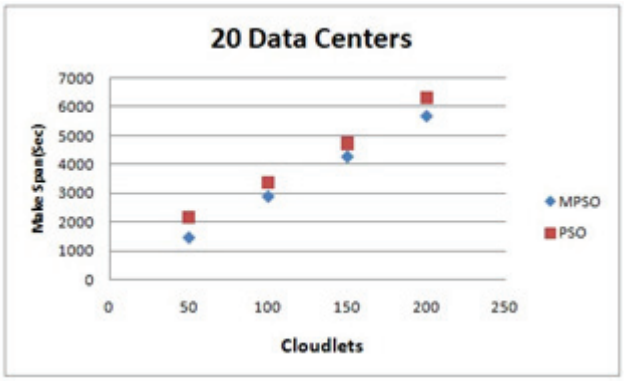

Fig. 4. PSO and MPSO makespan time for 20 Data Centers

Figure 4 shows the value of makespan is calculated for 20 data centers with each 50, 100, 150 and 200 cloudlets for both MPSO and PSO approaches.

C. Experimental Result

Comparison of existing Particle Swarm optimization is done with Proposed Mutation Based Particle Swarm Optimization. It is observed that the proposed technique gives an optimized MakeSpan rather than the existing PSO. It is observed that MakeSpan is nearly 68.23% more optimum in MPSO than existing PSO for 50 cloudlets, 86.78% for 100 cloudlets, 90.05% for 150 cloudlets and 91.32% for 200 cloudlets.

The comparison of the datacentres (DCs) for a varying number of cloudlets for the proposed algorithm (MPSO) is shown in Fig.5.

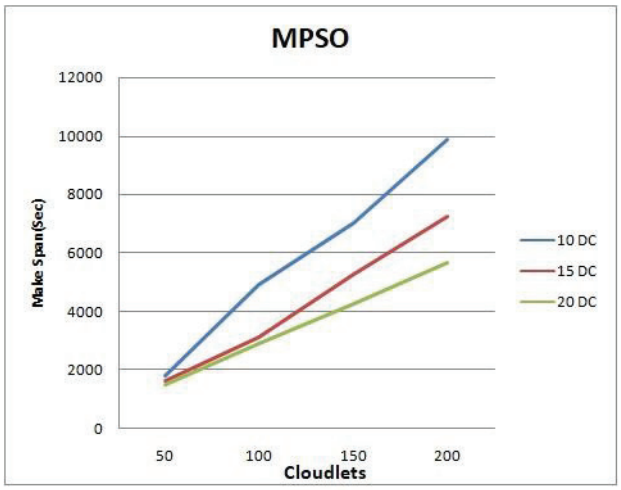

Fig. 5. PSO and MPSO makespan time for 10, 15 and 20 Data Centers

Figure 5 shows the comparison of the data-centers for varying number of cloudlets (50,100,150 and 200 cloudlets) with the value of makespan is calculated for 10, 15, and 20 data centers.

TABLE I. COMPARISON OF PSO AND MPSO MAKESPAN TIME FOR 10, 15 AND 20 DATA CENTERS

Data Centers	Avg. Makespan time of PSO (in a sec)	Avg. Makespan time of MPSO (in a sec)	Decreased Makespan time (in a sec)
10	6316.027	5900.859	415.168
15	4705.335	4332.470	372.865
20	4144.739	3585.25	559.489

Table I shows the comparison of PSO and MPSO makespan time. In an experiment with 10 datacentres avg. The makespan time of PSO is 6316.027sec while avg. The makespan time of MPSO is 5900.859sec, which decreases makespan time with 415.165sec. In an experiment with 15 datacentres avg. The makespan time of PSO is 4705.335sec while avg. The makespan time of MPSO is 4332.470 sec, which decreases makespan time with 372.865sec. In an experiment with 10 datacentres avg. The makespan time of PSO is 4144.739sec while avg. The makespan time of MPSO is 3585.25sec which decreases makespan time with 559.489sec.

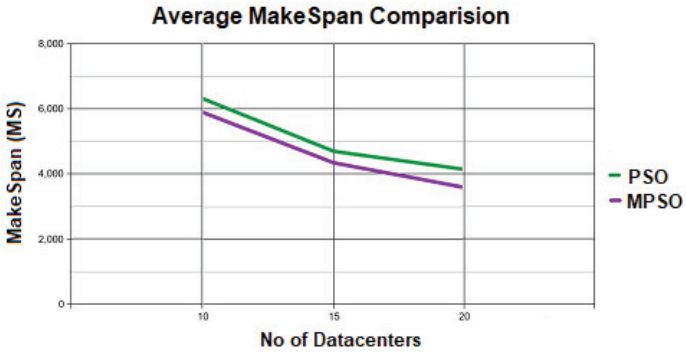

Fig. 6. The comparison of average Makespan

Figure 6 shows the comparison of average Makespan values calculated for existing PSO and proposed model MPSO. The costs are compared for a different number of data centers and the number of cloudlets is also varied accordingly.

The comparison of average Makespan is done for existing PSO and proposed model MPSO. The average difference thus found out is nearly constant for all values. The result is validated as the makespan of the proposed MPSO is lesser than the existing PSO. The fitness function is also improved.

IV. CONCLUSION AND FUTURE WORK

Load Balancing has always been the most significant challenge in the Cloud Computing environment. The primary objective of the proposed algorithm is to minimize MakeSpan and improve fitness function. Here proposed method uses a mutation-based particle swarm optimization. It is a nature-inspired optimization technique, so the proposed method combined it with a genetic operator i.e. Mutation. Applying mutation on the best solution given by the existing PSO algorithm gave us a better MakeSpan and improved the fitness function. MPSO algorithm gives better results as compared to PSO. Here proposed method is also compared on the MakeSpan parameter. The proposed method used pre-emptive virtual machine scheduling for performing load balancing. In the future, various other metrics like throughput, average time, resource utilization, waiting time, etc. can be considered.

REFERENCES

[1] J. Acharya, M. Mehta, and B. Saini, "Particle swarm optimization based load balancing in cloud computing," in 2016 International Conference on Communication and Electronics Systems (ICCES), pp. 1–4, IEEE, 2016.

[2] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A genetic algorithm (ga) based load balancing strategy for cloud computing," Procedia Technology, vol. 10, pp. 340–347, 2013.

[3] I. Nwobodo, "Cloud computing: A detailed relationship to grid and cluster computing," International Journal of Future Computer and Communication, vol. 4, no. 2, p. 82, 2015.

[4] M. Mesbahi and A. M. Rahmani, "Load balancing in cloud computing: a state of the art survey," Int. J. Mod. Educ. Comput. Sci. Vol. 8, no. 3, p. 64, 2016.

[5] N. J. Kansal and I. Chana, "Existing load balancing techniques in cloud computing: A systematic review," Journal of Information Systems and Communication, vol. 3, no. 1, p. 87, 2012.

[6] A. A. S. Farrag, S. A. Mahmoud, and M. El Sayed, "Intelligent cloud algorithms for load balancing problems: A survey," in 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS), pp. 210–216, IEEE, 2015.

[7] S. Bareen, K. Shinde, and S. Borde, "Challenges of big data processing and scheduling of processes using various hadoop schedulers: a survey," International Journal of Multifaceted and Multilingual Studies, vol. 3, no. 12, 2017.

[8] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in Proceedings of the 5th European conference on Computer systems, pp. 265–278, ACM, 2010.

[9] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with mapreduce: a survey," AcM sIGMoD Record, vol. 40, no. 4, pp. 11–20, 2012.

[10] A. Nait-Ali, "Proceedings of the 2nd ieee international conference on bio-engineering for smart technologies, biosmart 2017," 2016.

[11] M. Yakhchi, S. M. Ghafari, S. Yakhchi, M. Fazeli, and A. Patooghi, "Proposing a load balancing method based on cuckoo optimization algorithm for energy management in cloud computing infrastructures,"

in 2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO), pp. 1–5, IEEE, 2015.

[12] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, R. Rastogi, et al., "Load balancing of nodes in cloud using ant colony optimization," in 2012 UKSim 14th International Conference on Computer Modelling and Simulation, pp. 3–8, IEEE, 2012.

[13] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, "Load-balancing algorithms in cloud computing: a survey," Journal of Network and Computer Applications, vol. 88, pp. 50–71, 2017.