An Efficient Load Balancing for Cloud Computing Environment Using an Enhance Particle Swarm Optimisation Algorithm

Okere Chidiebere Emmanuel[1],Mustapha Lawal Abdurlrahman[1], Muhammad Lamir Isah[2], Fatima Umar Zambuk[1], Fatima Shittu[1], Goteng Kuwindi Job[1]
[1]Department of Mathematical science Abubakar Tafawa Balewa University Bauchi, Nigeria.
Corresponding Author:Okere Chidiebere Emmanuel

Submitted: 10-08-2021 Revised: 22-08-2021 Accepted: 25-08-2021

ABSTRACT: Task scheduling is the most important requirement on a cloud as it plays the key role of ensuring that the whole cloud computing facilities are used efficiently. Task scheduling ensures that best suitable resources required for a task to be executed are provided so that efficiency can be achieved with respect to different performance metrics like time, cost, scalability, make span, reliability, availability, throughput, resource utilization and so on. The proposed algorithm's design is anchored on reliability and availability. Because achieving these performance metrics is complex, a mathematical model was proposed for load balancing. This goal is to balance a particle swarm through efficient scheduling of tasks and adequate resource allocation by taking into account the following parameters: reliability, execution time, transmission time, make span, round trip time, transmission cost and load balancing between tasks and virtual machine. The proposed algorithm can play a role in achieving reliability of a typical cloud computing environment. The proposed method was compared with standard PSO, and Longest Cloudlet to Fastest Processor (LCFP) algorithm and results show that the Proposed PSO-based algorithm is efficient with respect to Makespan, Average Waiting Time, Average Response Time, and Time Complexity.

I. INTRODUCTION

The National Institute of Standards and Technology (NIST) defined the Cloud as "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." The Cloud is a shared infrastructure which can attach huge pools of systems in order to provide users with a variety of storage and computing resources via the internet on rental basis. Its major potential is the provision of Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS) [1]. Despite the advantages that the Cloud provide, it comes with it the problem of load imbalance. This problem is faced in two major areas: Task Scheduling and Virtual Machine Placement [2]. Task scheduling, the most troubling problem is concerned with search for optimal schedules in view of a number of constraints. Task scheduling entails optimal usage of scarce resources. A high-performance cloud is one that is efficient in resource scheduling. Most existing load balancing algorithms do not consider reliability and availability which should be considered. Load balancing on a cloud is an NP-hard problem. PSO as a heuristic algorithm has been used greatly in solving load balancing and other NP-hard problems. This research aims at enhancing the overall performance of PSO algorithm. The proposed method is however based on the PSO algorithm. This algorithm is proposed to achieve reliability with respect to task scheduling by taking into account the following parameters: Makespan, Average Waiting Time, Average Response Time, and Time Complexity.

II. PROPOSED METHOD

2.1 Benchmark Algorithm

Particle Swarm Optimization (PSO) [3] is a computational method that optimizes a problem by iteratively improving a candidate solution with regard to a given measure of quality. A problem is solved using PSO by having a population of candidate solutions (called particles), and then by moving these particles around in the search-space using simple mathematical formulae over the particle's position and velocity. The movement of each particle is influenced by its known position, but is guided toward the best-known positions in the search-space, which are updated as better positions are found by other particles. This approach is expected to move the swarm (of particles) toward the best solutions.

PSO is a metaheuristic approach [3]as it makes no assumptions about the problem being optimized and is capable of searching very large spaces of candidate solutions. However, metaheuristic algorithms such as PSO do not guarantee that an optimal solution is ever found. Also, the PSO algorithm does not use the gradient of the problem being optimized, which means the PSO algorithm does not require that the optimization problem be differentiable as it is in the case of some classic optimization methods such as gradient descent and quasi-newton methods.

Formally speaking, let f: $\mathbb{R}^n \rightarrow \mathbb{R}$ be the cost function to be minimized. The cost function takes a candidate solution (or a particle) as an argument which is in the form of a vector of real numbers and then produces a real number as output. This output indicates the objective function value of the given candidate solution. As earlier stated, the gradient of objective function f is not known. The goal is to find a solution **a** for which f(**a**) ≤ f(**b**) for all **b** in the search-space, making **a** the global minimum. Maximization is achieved using the function h = -f.

So, let S be the number of particles (or candidate solutions) in the swarm, with 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. A basic PSO algorithm is then give as:

Table I: Particle Swarm Optimization (PSO) Algorithm

```
foreach particle i = 1, … , S do
    Initialize the particle’s position with a uniformly distributed random vector: $\mathbf{x}_i \sim U(\mathbf{b}_{lo}, \mathbf{b}_{up})$
    Initialize the particle's best known position to its initial position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
    if $f(\mathbf{p}_i) < f(\mathbf{g})$ then
        update the swarm's best known position: $\mathbf{g} \leftarrow \mathbf{p}_i$
        Initialize the particle's velocity: $\mathbf{v}_i \sim U(-|\mathbf{b}_{up}-\mathbf{b}_{lo}|, |\mathbf{b}_{up}-\mathbf{b}_{lo}|)$
while a termination criterion is not met, do:
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 $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ then
    Update the particle's best-known position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
    if $f(\mathbf{p}_i) < f(\mathbf{g})$ then
        Update the swarm's best-known position: $\mathbf{g} \leftarrow \mathbf{p}_i$
```

The values $\mathbf{b}_{lo}$ and $\mathbf{b}_{up}$ are respectively the lower and upper boundaries of the search-space. The termination criterion can be the number of iterations performed, or a solution where the adequate objective function value is found. The parameters ω, φ_p, and $\varphi_g$are selected by the practitioner and control the behavior and efficacy of the PSO method.

2.2 Proposed PSO-based Algorithm

Task Model

Definition 1 When a VM executes a task, the task consumes resources. This task resource consumption is directly proportional with the availability of resources at its respective node (VM).

T_i= [$Tcpu_i$, $Tmem_i$, $Tdisk_i$, $Tnet_i$]; (1)

Definition 2 Execution Time: For each task i to be executed on VM j, the time it will take to complete task i is known as its weight or execution time and is represented as E_i^j.

Load Model

Definition 3 Node Load: The load at a given node is the summation of the weight of all tasks allocated to such a node. This can be represented as L_i^j.

$L_i^j = \sum_{i=1}^{n} Ei^j$; (2)

Definition 4 Load Benchmark: This value is a computation of the average of Node Load L_i^j represented as benchmark(L) benchmark(L) = $\sum_{i=1,j=1}^{n,m} Li^j / m$; (3)

The Proposed PSO-based Algorithm maintains a (ACO) to organize VMs such that each node on the tree is a VM. Each node holds single or multiple task entries. Once tasks arrive, the tasks are allocated to the nodes (VMs) of the ACO by a task allocation algorithm. This process continues until all tasks on all nodes (VMs) have finished execution and deleted from nodes. The ACO helps manage the search space by ensuring that VMs to the left subtree of the ACO are underloaded ones while those to the right subtree are overloaded VMs. The proposed algorithm achieves load balancing by continuously migrating VMs to nodes that make the ACO balanced; that is, the left and right subtrees having loads that are almost equal. The proposed PSO-based algorithm is below:

```
Algorithm: Proposed PSO-based Algorithm for Load Balancing
Input: a set of Virtual Machines VMs
Output: a set of executed VMs on a balanced ACO

STEP 1: l_load = compute load of left subtree;
STEP 2: r_load = computer load of right subtree;
STEP 3:
        if l_load>r_load
                migrate largest VM in left sub-tree to right sub-tree
        else if l_load<r_load
                migrate largest VM in right sub-tree to left sub-tree
        else
               ACO is balanced
STEP 4: terminate when all VMs have finished execution
```

Fig. 1: Proposed PSO-based Algorithm for Load Balancing

2.3 Task Entry Structure and Virtual Machine Parameters

The Task Entry for every VM is a four tuple where id mean a unique task identification number, vm_id is a unique VM identification number, exectime refers to task execution time or weight, and comptime refers to estimated completion time for task.

T_{id} = <id, vm_id, exectime, comptime>; (4)

Every VM also maintains certain parameters to help monitor its activities. The parameters are vm_id which means VM unique identification number, load refers to the total weight of tasks allocated to the VM, finishtime refers to the timestamp the VM will finish executing the tasks allocated to it, and last refers to the time the last task finished execution.

VM_{id} = <vm_id, load, finishtime, last>; (5)

2.4 Task Allocation Algorithm

```
Algorithm: Task Allocation Algorithm
Input: a set of Tasks T
Output: a set of  VMs placed on a balanced ACO

STEP 1: If ACO is empty, create new VM and insert on the ACO.
STEP 2: Identify VM with the least load on the ACO and assign task to it. Repeat this process
until all tasks have been allocated.
STEP 3: While VMs are executing tasks, use the PSO algorithm to optimize solution based on
indicated performance metrics.
STEP 4: Terminate algorithm when VMs have executed all tasks.
```

Fig. 2: Task Scheduling Algorithm

III. RESULTS AND DISCUSSION

3.1 Experimental Conditions

CloudSim was used to model and simulate VMs, computing resources, and energy consumption in order to evaluate the efficiency of load balancing for the Proposed PSO-based Algorithm. MATLAB 2018a was used to simulate the Proposed PSO-based (P-PSO) Algorithm, the standard PSO (S-PSO) algorithm, and the Longest Cloudlet to Fastest Processor (LCFP) [4] algorithm. This paper compares the performance of these algorithms using the following performance metrics: Makespan, Average Waiting Time, Average Response Time, and Time Complexity. In this experiment, 20 tasks were scheduled against 6 VMs. Table I presents the 20 tasks and the execution time while Table II presents efficiency grade for proposed PSO-based algorithm, standard PSO algorithm, and Longest Cloudlet to Fastest Processor algorithm respectively. Fig. 3 pictorially describes the result of the experiment.

Table 3: Tasks and their Execution Time

Task ID	Execution Time (in Seconds)
1	10.2
2	23.5
3	14.1
4	15.6
5	17.3
6	21.9
7	14.0
8	19.6
9	15.2
10	12.5
11	17.1
12	22.8
13	16.2
14	14.6
15	12.1
16	23.2
17	13.9
18	20.5
19	22.3
20	19.6
21	21.9
22	14.0
23	19.6
24	15.2
25	12.5
26	17.1
27	22.8
28	16.2
29	14.6
30	12.1
31	23.2
32	13.9
33	20.5
34	22.3
35	19.6
36	15.6
37	17.3
38	21.9
39	14.0
40	19.6
41	15.2
42	12.5
43	17.1
44	22.8
45	12.1

Table II: Efficiency Grade for Proposed PSO-based, Stand PSO, and LCFP algorithms

Algorithm	**Makespan**	**Average Time**	**Waiting**	**Average Response Time**	**Time Complexity**
P-PSO	0.219299	0.002321		0.021234	O(nlogn)
S-PSO	0.327217	0.012324		0.145432	O(n^2)
LCFP	0.638821	0.102321		0.211232	O(n^2)

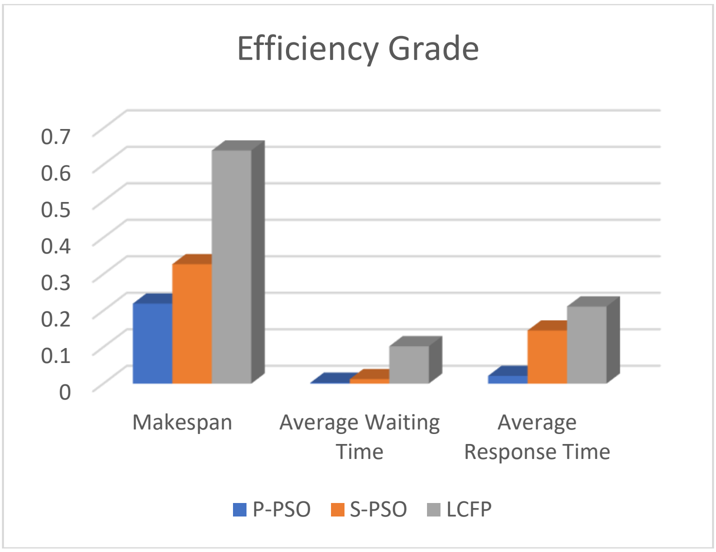

Fig 3: Efficiency Grade distribution of Proposed PSO-based, Standard PSO, and Longest Cloudlet to Fastest Processor

3.1 Discussion of Results

The result of this experiment (as shown in Fig. 3) is tied to the understandability and implementation of the comparative algorithms.

Table II and Fig. 3 shows clearly that the Proposed PSO-based algorithm outperforms Standard PSO and Longest Cloudlet to Faster Processor algorithms with respect to Makespan, Average Waiting Time, Average Response Time, and Time Complexity. From Table II and Fig. 3, it can also be seen that the Proposed PSO-based algorithm is efficient in 4 performance metrics which also means that the Proposed PSO-based algorithm is more reliable than its comparative algorithms.

In theory, this result is an evidence of advancements in the Load Balancing algorithms domain. In practice, the Proposed PSO-based algorithm will aid cloud providers improve on Makespan, Average Waiting Time, Average Response Time, and Time Complexity while it will in turn help them reduce Processing Cost and Energy Consumption.

IV. CONCLUSION AND FUTURE WORK

This paper introduced the 'Proposed PSO-based' algorithm. The results obtained after experimentation is a strong indicator that the Proposed PSO-based algorithm is more efficient than Standard PSO and Longest Cloudlet to Fastest Processor algorithms with respect to Makespan, Average Waiting Time, Average Response Time, and Time Complexity. The Proposed PSO-based algorithm does not put task migration into consideration which might leave some virtual machines overloaded. Further research is recommended in the area of task migration so as to improve on other performance metrics like Average Waiting Time and Average Response Time.

ACKNOWLEDGEMENT

We acknowledge the support of Department of Mathematical Science, Abubakar Tafawa Balewa University, Bauchi.

REFERENCE

[1]. H. Alexa and C. James, "The Basics of Cloud Computing," Carnegie Mellon University, 2011.
[2]. G. Chunye, L. Jie and Z. Qiang, "The Characteristics of Cloud Computing," in 2010 39th International Conference on Parallel Processing Workshops, San Diego, CA, USA, 2010.
[3]. L. Zhanghui and W. Xiaoli, "A PSO-Based Algorithm for Load Balancing in Virtual Machines of Cloud Computing Environment," in International Conference in Swarm Intelligence, Berlin, 2012.
[4]. S. Sindhu and M. Saswati, "Efficient Task Scheduling Algorithms for Cloud Computing Environment," in High Performance Architecture and Grid Computing: International Conference, Changiagarh, India, 2011.
[5]. Torry Harris, "Cloud Computing - An Overview," [Online]. Available: http://www.thbs.com/downloads/Cloud-Computing-Overview.pdf. [Accessed 21 December 2017].
[6]. G. Yongqiang, G. Haibing, Q. Zhengwei, H. Yang and L. Liang, "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.
[7]. K. C. N. M. Mosharaf and B. Raouf, "A survey of network virtualization," Computer Networks, vol. 54, no. 5, pp. 862-876, 2010.
[8]. B. Paul, D. Boris, F. Keir, H. Steven, H. Tim, H. Alex, N. Rolf, P. Ian and W. Andrew, "Xen and the art of virtualization," in SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles, 2003.
[9]. L. Flavio and D. P. Roberto, "Secure virtualization for cloud computing," Journal of Network and Computer Applications, vol. 34, no. 3, pp. 1113-1122, 2011.
[10]. L. K. Ronald and D. V. Russel, Cloud Security: A Comprehensive Guide to Secure Cloud Computing, Wiley Publishing, 2010.
[11]. J. Y and M. K, "Cloud computing - concepts, architecture and challenges," in Computing, Electronics and2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET), Kumaracoil, India, 2012.
[12]. A. Mohammad and H. Eui-Nam, "Fog Computing and Smart Gateway Based Communication for Cloud of Things," in 2014 International Conference on Future Internet of Things and Cloud, Barcelona, 2014.
[13]. Z. Qi, C. Lu and B. Raouf, "Cloud computing: state-of-the-art and research challenges," Journal of Internet Services and Applications, vol. 1, no. 1, pp. 7-18, 2010.
[14]. K. Dzmitry, B. Pascal and U. K. Samee, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," The Journal of Supercomputing, vol. 62, no. 3, pp. 1263-1283, 2012.
[15]. K. M. Nitin and M. Nishchol, "Load Balancing Techniques: Need, Objectives and Major Challenges in Cloud Computing - A Systematic Review," International Journal of Computer Applications, vol. 131, pp. 11-19, 2015.
[16]. R. Martin, L. David and T.-B. A, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing," in 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, Perth, WA, Australia, 2010.
[17]. W. Lee, J. S. Howard, P. R. Vwani and A. M. Anthony, "Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach," Journal of Parallel and Distributed Computing, vol. 47, no. 1, pp. 8-22, 1997.
[18]. S. Pinal, "A SURVEY OF VARIOUS SCHEDULING ALGORITHM IN CLOUD," International Journal of Research in Engineering and Technology, vol. 2, no. 2, pp. 131-135, 2013.
[19]. M. M, M. N, K. Y, C. L. Y, G. T. E, Y. Z. A and T. D, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," Journal of Parallel and Distributed Computing, vol. 71, no. 11, pp. 1497-1508, 2011.
[20]. F. B. Luiz and R. M. M. Edmundo, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," Journal of Internet Services and Applications, vol. 2, no. 3, pp. 207-227, 2011.
[21]. T. M. Siva, S. R and Y. Lei, "Stochastic models of load balancing and scheduling in cloud computing clusters," in 2012 Proceedings IEEE INFOCOM, Orlando, FL, USA, 2012.
[22]. S. Pooja and M. Pramati, "Analysis of variants in Round Robin Algorithms," International Journal of Computer Science and Information Technologies, vol. 4, no. 3, pp. 416-419, 2013.

[23]. M. Yinchi, C. Xi and L. Xiaofang, "Max-Min Task Scheduling Algorithm for Load Balance in Cloud Computing," in Proceedings of International Conference on Computer Science and Information Technology, Advances in Intelligent Systems and Computing 225, India, 2014.
[24]. B. K. Remesh and P. Samuel, "Enhanced Bee Colony Algorithm for Efficient Load Balancing and Scheduling in Cloud," Innovations in Bio-Inspired Computing and Applications, vol. 424, pp. 67-78, 2016.
[25]. M. Gao and L. Hao, "An Improved Algorithm Based on Max-Min for Cloud Task Scheduling," Recent Advances in Computer Science and Information Engineering, vol. 125, pp. 217-223, 2012.