# Hybrid Metaheuristic Algorithm & QoS in Cloud Computing Environment

**Gold Mine[1], Jaspal Singh[2]**
[1]Dept of Computer Science and Engineering
[2]Assistant Professor, Dept of Computer Science and Engineering
[1, 2] SLIET Longowal, India

***Abstract-*** *The attractive features of Cloud Computing (CC) has become popular. With the development of new applications, the cloud load increases tremendously. Load Balancing (LB) is an important part of the Cloud Computing environment that ensures that all devices or processors work at the same time. Various LB models and algorithms have been developed in CC to make cloud resources easily accessible to the end-users. In our research, we used HIDE for cloud computing to tackle Load Balancing. To increase the performance of an algorithm, a dynamic pitch-adjusting rate has also been implemented in the terms of QoS like cost based on service. Our work is more relevant than the Previous work is being conducted in less time with the clouds and more effectively perform LB.*

***Keywords***- Differential algorithm, Harmony search, Cloud computing, Task scheduling, Hybrid meta-heuristic.

## I. INTRODUCTION

The cloud computing is a computing standard that provides an on-demand, extremely manageable, elastic platform and robust with the collection of geographically separate information centers for multiple distributed computing applications. Also, the number and distinct kinds of VM resources can be given in clouds on demand for workflow execution. Due to advances in mobility, the hit rate of internet applications such as Google, Facebook, Amazon, etc. is increasing exponentially.

According to "Cloud Computing" all programs and their data required for the remote server on the Internet are initiated and output results of research in the normal web browser window on a local PC. The cloud computing advantages include a reduction in computing power requirements of PCs, improved FT (Fault Tolerance) & protection, often increasing data processing speed, the expense of software and hardware decreased power, decreased repair costs, and saved disk space.

The load balancing is the process of changing the workload between the processors to improve system performance. The workload of a machine refers to the total time it takes for a machine to perform all tasks[1]. The load balancing is done to increase throughput & minimize response time for each VM (Virtual Machine) in the cloud system throughout the entire time. LB is one of the key factors for increasing cloud service providers' working performance. VM load balancing ensures that no machine is idling or partly loaded during the fast loading of other machines. The dynamic allocation of workload is one of the major issues of cloud computing. The advantages of distributing the workload include an improvement in the use of resources, which further improves overall efficiency and achieves optimum customer satisfaction.

## WHY BALANCING IN CLOUD COMPUTING?

The load balancing is an extremely complex local workload spread uniformly over all nodes in clouds. It is utilized to maintain resource utilization ratio and high user satisfaction [3], meaning that the system's total output is not overshadowed by a single node. Proper load balancing will allow efficient use of available resources, thereby reducing the consumption of resources. It also helps to manage failure, enable scalability, reduce response time, prevent bottlenecks & over-supply, etc. Load Balancing is also important to achieve green cloud computing.

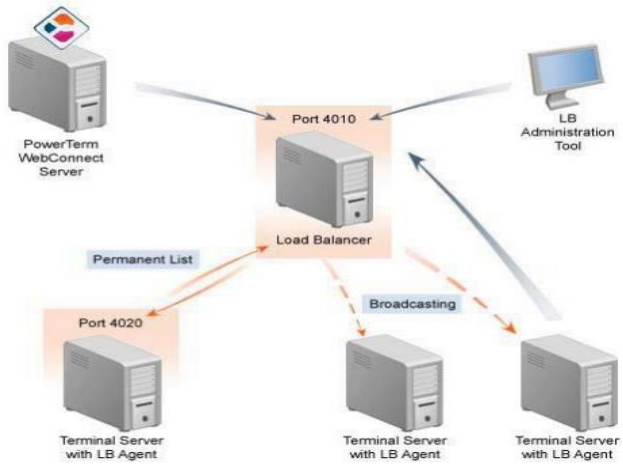


**Figure 1: Load balancing in cloud computing**

*Factors responsible for it are:*

• Limited Energy Consumption: The load balancing can reduce energy consumption by preventing undue workload due to the center of nodes or virtual machines.

• Reducing Carbon Emission: 2 sides of a similar coin are coal consumption and carbon emissions. They are also strictly related. Load Balancing helps minimize energy consumption and reduces carbon emissions automatically, which thus makes Green Computing possible.

**LOAD BALANCING: It's a GOAL**

Goals of Load Balancing are:
• To have a backup plan in case the system faces even a small failure
• To enhance performance considerably
• Accommodation of future modification in a system
• To maintain system stability.

## II. RELATED WORK

The related research and task scheduling are discussed in this section. The job schedule is to ship jobs to optimal resources and can be split into several activities. The role of Load Balancing is to balance the load that is distributed between workers and optimal resources. The Load Balancing working in the geo-distributed cloud currently focuses on raising the workload time and maintaining the device load balance.

Different methods have been suggested in various experiments using the methodology of clustering and logging, but also some limitations. In [4] the authors propose to place it at any Data Center (DC), using the K-Means approach depends upon clustering VMs. In the clustering algorithm, an attribute is only the RAM size of the VM. They utilize a mathematical model to illustrate the proposed system concept.

The proposed architecture of authors in [5] uses clustering technologies for the distribution of cloud resources. They utilize Google's cluster tracks to build task groups and project them out to virtual machines, taking into consideration each cluster's real resource usage and do not rely on the services the users are looking for. The method of mapping is focused on patterns of task usage got from historical data.

In [6], the authors suggest a Data Center web-business development model for cloud loading to extract behavioral patterns of individual users and to help network analyzes and simulation phases. They are using graphical and statistical model testing models, but the model does not accept any constraints such as consumer arrival and daily cycle characteristics. Furthermore, the model does not determine the effect on large metrics of various sizes of users. Authors examine that user behavior in workload modeling should be assumed to reflect actual situations, provided that the form of user profile has a significant effect on the use of resources.

In [7], a dynamic VM allocation policy is developed by the use of clustering techniques to set up groups of VMs. In compliance with user specifications, this approach takes the VMs and generates VMs using K-Means, which are then transmitted for delivery to the closest accessible data center. In [8], the Google Monitoring Data Collection is used to define and coordinate tasks depend upon the usage of tasks across people. The proposed method aims to the placement of VMs allocated by clusters on the same hosts.

In [8], suggested throttled Load Balancing algorithm that reduces cost & response time in multi-datacenter VMs as well as optimizes response time. The algorithm described in this paper was that the customer first requests a load balance system to find an adequate virtual machine that can operate the incoming process. Several virtual machine instances can occur in cloud computing. These VMs may be clustered as per the type of request. It works correctly according to the incoming requests. When a customer transmits a request, the load balancer firstly checks the package as well as will delegate the request when he is ready to accept & process the request.

In [9], the suggested min-min algorithm, that chooses maximum completion time task as well as assigns to its suitable resource for improved exploitation and resource management. When no. of tasks in meta-task exceeds no. of big tasks, the Min-Min algorithm does not adequately schedule tasks; as well as the structure of the scheme is comparatively large. Moreover, a load-balanced system does not provide. This algorithm provides higher priority to small tasks also increases the time for large tasks to respond. Therefore, this algorithm's downside is that several jobs can experience starvation.

In [10], the proposed stochastic hill-climbing algorithm i.e. utilized to delegate incoming jobs to servers or VMs. Cloud Analyst analyzes the algorithm's performance qualitatively & quantitatively. Cloud observer is a cloud-based virtual modeler for web device and application analysis. In this paper, the author explains the method in the cloud computing environment for load distribution. The downside is that in many situations this strategy has not been properly optimized.

## III. RESEARCH METHODOLOGY

In this part, we illustrate our method to involuntarily make VMs group in cloud datacenter depends upon their source custom use and to involuntarily make users task groups depend upon their resource requirements.

**Problem Definition:** The Load Balancing goal is to identify the needs of customers simply and without delay data and information can be sent and received. The Load Balancing in cloud storage is one of the main challenges because it can be a network, memory because CPU load without delay while load balancing applications. The Load Balancing can have time-consuming system responses. Cloud computing provides a comprehensive array of shared tools, software packages, information, storage, and various applications at any time according to user demands. Better Load Balancing in the cloud network enhances the efficiency and usage of resources through the dynamic distribution of workload across different cloud nodes.

***Proposed Model:***

To overcome the above problems, we have proposed a new methodology which is "Harmony Inspired Differential Evolution Algorithm"(HIDE) which tends to the problem of existing research technique and provide the overwhelming result by sorting out the problems of the Genetic algorithm which was applied in the previous research methodology.

**HARMONY SEARCH ALGORITHM:**

HS model is based on the principle of swarm intelligence, suggested in the optimization of particle swarm [1], in which the current harmony would memorize the knowledge from the harmony that has been strongest. To increase the performance of an algorithm, a dynamic pitch-adjusting rate[8] has also been implemented. At GHS, three methods were used to create a new harmony: memory analysis, global pitch adjustment, and random search.

**Random Search:** Search space spontaneously generates the components of the current harmony as follows:

$$x_j^{new} = a_j + r(b_j - a_j), j = 1, \dots, n \quad (1)$$

where $x_j^{new}$ is the j[th] element of new harmony, as well as r, is a real number consistently partitioned in [0,1].

**Memory Consideration:** Novel harmony components are unorderly selected from harmony memory is shown here:

$$x_j^{new} = x_j^m, j = 1,\dots n \quad (2)$$

where $x^m$ is harmony partially selected from HM

**Global Pitch Adjustment (GPA):** Memory vector is passing with a PAR(Pitch Adjusting Rate) likelihood on its neighbor by imitating the finest harmony in Harmony Memory (HM):

$$x_j^{new} = x_k^{best}, j = 1\dots.n \quad (3)$$

Where $x^{best}$ represents worth harmony in HM also $k$ is numeral randomly selected amid 1 & $n$

In GHS, 2 parameters HMCR (Harmony Memory Considering Rate) & PAR(Pitch Adjusting Rate) being utilized for managing the ratio of various methods, these 2 characters help searching technique for widely and locally elucidation, correspondingly [10]. Random search is utilized in GHS with a probability of 1 and with an HMCR probability of HMCR. GPA is used over components created using the PAR likelihood of memory, which decreases linearly in the searching procedure:

$$PAR(t) = \mathrm{PAR}_{\min} + \frac{(PAR_{max} - PAR_{max}}{NI} t \quad (4)$$

where *NI* means the total number of increments. The major process of GHS starts, where $f_i$ represents function value of harmony *i*, *n* represents decision vector dimension, *U(0,1)* represents static partition amid 0 & 1. Pseudocode 1 is a stepwise demonstration of HSA (Harmony Search algorithm).

**Pseudocode HAS**

1: In favor of every Harmony in HM:
1.1: Randomly input $X_i$ as per (2);
1.2: Calculate $f_i$ ;
1.3 Initiate $X^{best}$ with the worth harmony.

2: For all magnitude*i*($i \in [1,n]$) :
2.1: Change *PAR(t)* as per (4);
2.2: Produce $x_j^{new}$ as follows:
**If** $U(0,1) \leq HMCR$ **then**
Memory management as per (2);
**If** $U(0,1) \leq PAR(t)$ **then**
GPA as per (3);
**Endif**
**Else**
Random electionas per (2);
**Endif**
Step 3: New harmony will change unuseful harmony in HM i only when it's improved.

Step 4: Go back to step 2 if the stopping criterion is not satisfied, else stop.

**DIFFERENTIAL EVOLUTION (DE):**

DE is population dependent stochastic global optimizer capable of maintaining a non-linear & multimodal environment. DE performs several additional optimization techniques in terms of convergence as well as stability over general benchmark & real-world applications. Many variants have been suggested for the classical DE. DE represented as *DE/rand/1/bin* is accepted into this work. Usually speaking, DE power may be recognized to 4 mechanisms: differential operator, greedy selection, parent choice & discrete crossover.

**Parent choice:** Both people in the community have a fair chance of developing candidates. For each Xi individual, three more individuals from the present population are randomly selected, such that four individuals vary. A pool of 4 parents has therefore been established.

**Differential Operator:** An individual $V_i$ is produced after a mutation from 3 individuals in orderly selected in origin choice as follows:

$V_i = X_{r1} + F(X_{r2} - X_{r3})$ (5)

where *r1,r2,r3(r1≠ r2 ≠ r3 ≠ i)* are 3 in orderly elected index of inhabitants. As well as F is +ve real number which is often less than 1.0.

**Discrete Crossover:** A testing individual $T_i$ is created by elements combination of $X_i$ and $V_i$ one by one as follows:

$$t_{i,j} = \begin{cases} v_{i,j} \text{ if } r \leq cr \text{ or } j = sn \\ x_{i,j} \text{ otherwise} \end{cases} \quad j = 1, \ldots . n \qquad (6)$$

where $t_{i,j}$is $j$ the element of trial entity $T_i$ , $r$ is a static odd number in [0,1], $cr$ is an original number in [0,1] that manages the proportion of selection among inherent and vector after mutation. As well as $sn$ is an arbitrarily selected index to make sure that a minimum of 1component of $V_i$ will be produced.

**Greedy Selection:** It is applied between trial vector $T_i$ & $X_i$, as well as one with worth fitness function, will be encouraged to the next level.

The process of DE is shown in pseudocode 2, where $f_i$ represents the functional cost of individual *i*.

**Pseudocode DE**

1: For each entity in population:
1.1: Load $X_i$ randomly;
1.2: Calculate $f_i$ ;
1.3:Obtain the best entity $X^{best}$ ;
2: For each entity $X_i$ in population:
2.1: Randomly select 3values *r1,r2,r3*, such that $r1 ; \neq r2 \neq r3 \neq I$;
2.2: Get mutation value $V_i$ as per (5);
2.3: Get trial value $T_i$ as per (6);
2.4: Greedy selection between $T_i$ & $X_i$.
3: Return to step 2 if the above criteria are not satisfied, else terminate.

**ALGORITHM:**

**Step 1.** Begin
**Step 2.** Choose process
**Step 3.** Arrange process as per cost (cost based on service)
**Step4.**If ((Distance && resource requirement && process length) <= Threshold value)
// Let threshold value =35% of cost effective procedure for VM
**Else**
Costly process // search subsequent vm
**Step 5.** Now opt for shortlisted VM as well as process them by applying Harmony Inspired Differential Evolution
**Step 6.** Generate chromosome // using process & VM
**Step 7.** Apply crossover over chromosomes
**Step 8.** If (the distance of VM and process < threshold)
If (VM have sufficient space for process run)
Process assign to VM
**Else**
Search next VM for process
**Else**
Not sufficient
**Step 9.** Repeat step from 2 to 8, all new arrival process
**Step 10.** Exit

## IV. PERFORMANCE EVALUATION

In this paper, we have worked on 10 maximum virtual machines and 20 cloudlets. The implementation is done on a cloud simulator tool.

**Table 1: Configuration settings**

| | |
|---|---|
| Tool | Cloudsim |
| Vm | 10 |
| Cloudlets | 20 |
| Technique | HIDE |

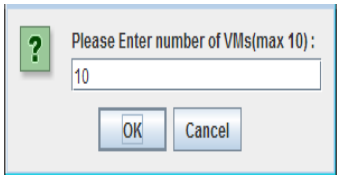


**Figure 2: No. of VMs selected**

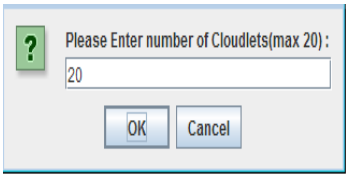


**Figure 3: No. of Cloudlets selected**

```
<terminated> TestGA (1) [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (Jul 23, 2020, 11:03:39 AM)
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet_ID     STATUS  DataCenter_ID   VM_ID   Time    Start_Time      Finish_Time
0               SUCCESS         2       8       0.91    0.1             1.01
11              SUCCESS         2       3       1.1     0.1             1.2
10              SUCCESS         2       8       1.21    0.1             1.31
5               SUCCESS         2       7       1.33    0.1             1.43
1               SUCCESS         2       3       1.55    0.1             1.65
9               SUCCESS         2       0       2       0.1             2.1
4               SUCCESS         2       1       2.5     0.1             2.6
12              SUCCESS         2       6       2.84    0.1             2.94
6               SUCCESS         2       9       2.95    0.1             3.05
3               SUCCESS         2       2       3.53    0.1             3.63
13              SUCCESS         2       2       4.7     0.1             4.8
19              SUCCESS         2       0       5       0.1             5.1
16              SUCCESS         2       9       5.11    0.1             5.21
8               SUCCESS         2       4       5.45    0.1             5.55
14              SUCCESS         2       1       5.67    0.1             5.77
15              SUCCESS         2       7       6       0.1             6.1
17              SUCCESS         2       5       6.67    0.1             6.77
2               SUCCESS         2       6       6.97    0.1             7.07
7               SUCCESS         2       5       8.33    0.1             8.43
18              SUCCESS         2       4       20.91   0.1             21.01

Finish Time 21.008181818181818
```

**Figure 4: Cloudsim showing result of GA**

```
<terminated> HideTest [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (Jul 23, 2020, 11:28:19 AM)
Broker1 is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet_ID     STATUS  DataCenter_ID   VM_ID   Time    Start_Time      Finish_Time
11              SUCCESS 2               4       1       0.1             1.1
0               SUCCESS 2               9       1.47    0.1             1.57
5               SUCCESS 2               9       1.47    0.1             1.57
10              SUCCESS 2               8       2.73    0.1             2.83
13              SUCCESS 2               6       2.84    0.1             2.94
16              SUCCESS 2               2       2.94    0.1             3.04
9               SUCCESS 2               0       3.05    0.1             3.15
6               SUCCESS 2               8       3.41    0.1             3.51
14              SUCCESS 2               3       3.52    0.1             3.62
3               SUCCESS 2               8       4.32    0.1             4.42
17              SUCCESS 2               8       4.77    0.1             4.87
1               SUCCESS 2               1       5       0.1             5.1
8               SUCCESS 2               5       5       0.1             5.1
4               SUCCESS 2               0       5.11    0.1             5.21
12              SUCCESS 2               0       5.56    0.1             5.66
19              SUCCESS 2               1       8.75    0.1             8.85
2               SUCCESS 2               5       10.83   0.1             10.93
7               SUCCESS 2               1       11.25   0.1             11.35
15              SUCCESS 2               1       12.5    0.1             12.6
18              SUCCESS 2               7       13.33   0.1             13.43

Finish Time             13.43
```

**Figure 5: Cloudsim showing result of HIDE**

**Table 1: Result visualization of GA and HIDE**

| Parameter | VM (I/P) | CLB (I/P) | Existing Technique finish-time GA | Propose Technique finish-time HIDE |
|---|---|---|---|---|
| 1. | 4 | 7 | 7.59 | 5.1 |
| 2. | 9 | 12 | 6.26 | 5.55 |
| 3. | 6 | 13 | 21.00 | 13.43 |
| 4 | 6 | 15 | 12.59 | 9.21 |
| 5 | 8 | 18 | 12.59 | 7.99 |

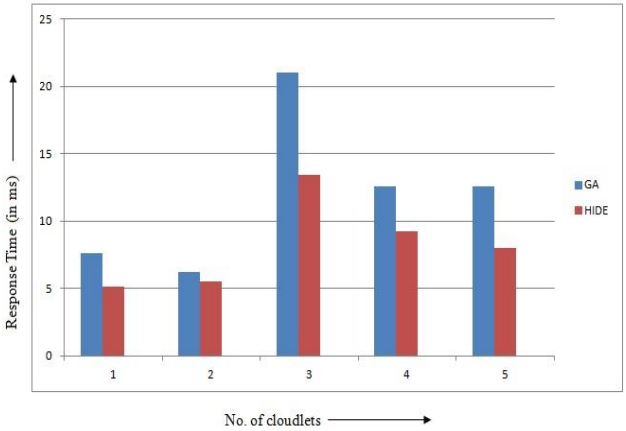


**Figure 6: A comparison chart of Base and Proposed methodologies**

## V. CONCLUSION

Due to the growing demand for high-performance computing resources, energy-efficient methods have recently been regarded to be of prime significance in cloud computing. Cloud Computing load balancing is a complicated issue of optimization that belongs to the NP-hard problem class. Meta-heuristic methods have proved to be an extremely efficient solution in this situation. We measured the issue of scheduling in heterogeneous virtualization of cloud environment wherever the number of tasks is scheduled on obtainable VMs by the goal of minimum energy consumption through preserving or civilizing the performance of cloud applications. By well-known nature-inspired Harmony Search algorithm and Differential Evolution algorithm, we have suggested a fresh hybrid strategy named HIDE.

HIDE offers rapid convergence by identifying whether or not the algorithm is in the local trap & removing the algorithm from a local trap, skipping iterations to decrease the performance period, and trying to move the worst alternatives from the finest alternatives. It combines the DE algorithm's exploration capacity with the harmony search algorithm's exploitation capacity with some extra characteristics associated with population enhancement. The

simulation experiments showed the efficiency of our technique in terms of less timing and more profitability.

## REFERENCES


[1] Chavan, V. (2014). "Clustered Virtual Machines for Higher Availability of Resources with Improved Scalability in Cloud Computing", IEEE.

[2] Piraghaj, S. F., Calheiros, R. N., Chan, J., Dastjerdi, A. V., &Buyya, R. (2017). "Virtual Machine Customization and Task MappingArchitecture for Efficient Allocation of Cloud Data Center Resources", 59(2), 208–224.

[3] Narkhede S., Baraskar T. Mukhopadhyay D. (2014), "Analyzing Web Application Log Files to Find Hit Count Through the Utilization ofHadoop MapReduce in Cloud Computing Environment". 978-1-4799-3064-7/14. IEEE.

[4] Piraghaj, S. F., Dastjerdi, A. V., Calheiros, R. N., &Buyya, R. (2015). "Efficient Virtual Machine Sizing For Hosting Containers as aService", 78-1-4673-7275-6, IEEE.

[5] Magalhães, D., Calheiros, R. N., Buyya, R., & Gomes, D. (2015). "Workload modeling for resource usage analysis and simulation in cloudcomputing", 47, 69–81.

[6] Adrian, B., & Computing, A. C. (2015). "Analysis of K-means Algorithm For VM Allocation in Cloud Computing", 48–53.

[7] Panchal, B., & Kapoor, P. R. K. (2013). Dynamic VM Allocation Algorithm using Clustering in Cloud Computing, 3(9), 143–150.

[8] Yousif, S. A., & Al-dulaimy, A. (2017). Clustering Cloud Workload Traces to Improve the Performance of Cloud Data Centers, I, 7–10.

[9] Ghomi, E. J., &Rahmani, A. M. (2017). "Load-balancing algorithms in cloud computing: A survey. Journal of Network and ComputerApplications", 88, 50–71.

[10] Prakash, P. G. O. (2016). "Analyzing and Predicting User Behavior Pattern from Weblogs", 11(9), 6278–6283.

[11] Singh, Jaspal & Goraya, Major. (2019). Multi-Objective Hybrid Optimization based Dynamic Resource Management Scheme for Cloud Computing Environments. 386-391. 10.1109/ICSSIT46314.2019.8987760.

[12] Thakur, A. (2017). PRIMARY ISSUES AND CHALLENGES IN CLOUD COMPUTING: A BRIEF NOTE. *International Journal of Advanced Research in Computer Science*, *8*(7), 436–438. https://doi.org/10.26483/ijarcs.v8i7.4274

[13] Yadav, Neeraj. (2019). A Ranking Based Model for Selecting Optimum Cloud Geographical Region. Journal of Engineering Design and Technology. 8. 793-797. 10.35940/ijitee.J8908.0881019.

[14] Neeraj, Goraya, M. S., & Singh, D. (2020). Satisfaction aware QoS-based bidirectional service mapping in cloud environment. *Cluster Computing*, *23*(4), 2991–3011. https://doi.org/10.1007/s10586-020-03065-7

[15] Agarwal, R., Baghel, N., & Khan, M. A. (2020). Load Balancing in Cloud Computing using Mutation Based Particle Swarm Optimization. 2020 International Conference on Contemporary Computing and Applications (IC3A). doi:10.1109/ic3a48958.2020.233295