**ISSN: 0975-766X**
**CODEN: IJPTFI**
***Available Online through*** ***Research Article***
**www.ijptonline.com**

# PERFORMANCE ANALYSIS OF ANT COLONY OPTIMIZATION AND GENETIC ALGORITHM FOR CLOUD LOAD BALANCING

**Senthil Murugan Balakrishnan[1] , Suresh Kumar Nagarajan[2]**
1Assistant Professor Senior, School of Information Technology and Engineering, VIT University, Vellore, Tamilnadu, India.
2Associate Professor, School of Computer Science and Engineering, VIT University, Vellore, Tamilnadu, India.
*Email: senthilmurugan.b@vit.ac.in*

*Received on 25-10-2016* *Accepted on 02-11-2016*

**Abstract:**

In this paper, we have analysed load balancing algorithms named as ACO and Genetic optimization, which are used for distribution of workload among the servers. Load balancing is major problem in cloud computing environment. We have proposed Load Balancing Unit which will be responsible to assign incoming tasks to appropriate server based on maximum capacity of each server. The main objective of our paper is to achieve Load balancing among various servers and optimize execution time of tasks. We have analysed performance of these algorithm based on parameters such as execution time, priority, cost and QoS. In ACO algorithm we are able to assign priority to important tasks and execute these tasks according to their priority. Providers must deal with load balancing problem otherwise client will switch for better service from another service provider. We compared these algorithms to get best load balancing solution based on execution time and cost. We try to give better QoS in peak usage hours based on analysis results.

**Key words:** ACO, Genetic optimization, Load Balancing Unit, Cloud Computing, QoS.

## 1. Introduction:

Cloud computing technology is adapted by most of the users across the world because of its service providing features. Cloud computing environment allows user to access computing resources at any time. Most features such as scalability, availability of resources are hard to achieve every time because of problem such as load balancing which is the major reason behind reduced QoS and increase in average waiting time. From provider point of view these are serious problems which can lead to reduction in number of users. In cloud computing environment clients don't' need to buy hardware or infrastructure to use it, client only have to pay according to time span for which they use particular service. Every Client is focus on how they can get required results as early as possible by reducing

execution time. To achieve maximum machine utilization the goal of these algorithms is that not allow any server to get over loaded by assigning tasks greater than its maximum capacity. In this approach we used Load Balancing Unit (LBU) who is totally responsible to manage assigning of tasks across the servers by checking their threshold value. We performed analysis on these algorithms to get best optimal solution for load balancing by increasing execution time of tasks because load on servers various rapidly in real time application scenario. There are two types of load balancing techniques one is static approach and another one is dynamic approach. Static approach suitable for small sets of incoming tasks and there is little variation among load. Static approach doesn't take current situation under consideration before making any decision while on other hand dynamic approach takes current status into account. To maintain updated information at each step leads to overhead drawback. Dynamic load balancing approaches are more superior in doing distribution of tasks as compared to static approach.

**2. Literature Survey:**

Kousik Dasgupta, Brototi Mandal and Paramartha Dutta[1] proposed a load balancing genetic algorithm (GA), which try to balance the cloud infrastructure load and reduce the execution time period of allocated task to machine so that maximum QOS can be achieved using efficient utilization of resources. To demonstrate how this algorithm better than other existing algorithms authors have used cloud analyst as a simulation tool to get results. All these results are compared with RR, FCFS, SHC on the basis of overall response time. Authors shown that this approach outperforms other algorithms.

**2.1 Honey bee behavior inspired load balancing (HBB-LB) algorithm**

Dhinesh Babu L.D, Venkata Krishna[2] proposed load balancing algorithm which is inspired from honey bee behavior, which is closely resemble to how honey bee search their food and do waggle dance to tell other bee in beehive about quality and quantity of food. In same way tasks from overloaded machines are treated as honey bees, these tasks are removed from overloaded machines and submitted to under loaded machines then task update the load of that particular machine for other waiting tasks. This will be useful for waiting tasks to choose virtual machine according to load conditions. Author used degree of imbalance to prevent dynamic load balancing. This algorithm clearly evident that after load distribution

**2.2 The Dual-Direction FTP Technique**

Nader Mohamed, Jameela Al-Jaroodi and Abdulla Eid[3] have demonstrated a load balancing technique which is called as dual direction scheme. In this technique they tried to make concurrent file transfer possible with less

overhead and improve file download from server. In this approach two DDFTP servers are used to download the file one of the DDFTP server starts downloading file blocks from left to right and another server start from right to left direction. The file transfer operation to client stops when both DDFTP servers meet. client doesn't need to do any kind of synchronization between two DDFTP servers. This approach allows client to fully utilize available bandwidth by receiving blocks simultaneously from multiple directions. They used reliable In order delivery feature of TCP to prevent it from any kind of error and coordination efforts of client. Automatic more work given to faster DDFTP server to cover more blocks than slower DDFTP server to achieve load balancing.

Authors compared DDFTP with regular FTP and DADTM that how this unique technique of parallelizing the download gives client effective bandwidth utilization and reduce RTT to get better performance.

**2.3 Ant Colony Optimization**

Ratan mishra, Anant jaiswal[4] proposed load balancing solution for cloud environment to minimize job response time. While moving from nest to food source ant drops pheromone liquid substance which can be followed by another ant that substance as a trail to reach source of food. Most of the trails consist of evident pheromone because one place over another by many ants. Same concept used by authors to find out destination node where load can be transferred to maintain load of network. In case of more than two trails ant takes path which has high pheromone concentration, in this algorithm authors maintained pheromone table which consist of node capacity, probability and response time of node. This helps to search destination node.

**2.4 Request balancing strategy:** Zeng Zeng, Bharadwaj[5] proposed balancing strategy which is known as Optimal metadata replication and request balancing strategy. In this paper all users request for retrieval of data submitted to data centre with request probability then file system portal chooses appropriate metadata server to transfer request of retrieval of information from raw server data. There may be more than one metadata server holding replicated copy of objects within same or different data centers. After receiving request from file system portal then metadata server gives command to raw data server to directly send object information to file system portal. In this paper author used Zipf law of distribution for distributing data with probabilities to respective metadata centres and achieve mean response time in high load cloud environment.

**2.5 Optimal load distribution**

Keqin Li[6]proposed solution for achieving less response time in cloud environment which is known as optimal load distribution mechanism. In this paper group of blade servers with each having its own size and speed managed by

load distribution. In this mechanism the work is distributed among two or more servers such that work gets done at same time and users gets response within short time. This technique uses thermo for special task which put into waiting queue ahead of normal task, in this way this mechanism provides load distribution for special task which required special attention and needs to be done immediately. Author found that server size, speed and arrival rate of special task affects average response time of normal task.

**3. Proposed System**

Proposed architecture is shown in Fig. 1. In this approach, we used both algorithms ACO and Genetic optimization to manage load distribution among servers. We modified ACO to minimize the pheromone table overhead at each node, In our proposed algorithm only Load Balancing Node contains updated pheromone table information. We used Euclidian distance to find out distance between head node and destination nodes to transfer load on the basis of threshold capacity to the shortest node.

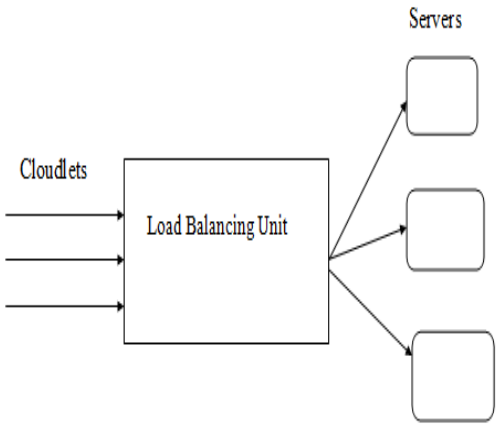


**Fig 1. LBU Architecture.**

Pseudo code Load Balancing using ACO

1 Initial pheromone = 0.8 d

2 Read pheromone = assign value from file

3 Initialize pheromone matrix

```
  for (column=0; columns < matrix.length; columns++)
{
   for ( int i=0; i < rows; i++)
    {
        local matrix [][] = initial pheromone;
```

```
    }
}
```

4 Calculate Euclidian Distance

```
Math.sqrt(math.pow(x2-   x1,2)+math.pow(y2-y1,2));
```

5 if (distance == 0)

```
Return 0;
Else return (1/distance);
```

6 Assign inverted value to local matrix

```
for(i=0; i <  matrix.length; i++){
        for(j=0;j < matrix.length; j++){
        local[][] = invert double(matrix[][]);
        } }
```

7 Best distance = way.distance;

After using ACO technique, we used Genetic optimization technique for balancing same tasks load and obtained results. Genetic technique consist of four steps

- Initial selection
- Crossover
- Fitness Function
- Mutation

In Initial selection process all available solutions for problems are converted into binary format. From this solution we randomly selected some solutions for next step. After finished with selection second step is crossover, whose main goal is to get minimal solution pair to crossover with each other. To find out best optimal solution among these solutions we have to use fitness function to calculate fitness value of each solution.

M = k1 * c (NIC / MIPS) + k2 * l

Where k1 & k2 are assigned weights to each solutions.

'c' is cost of execution and NIC is number of instructions present.

'l' is cost of delay we taken this value as constant for our implementation purpose.

Mutation step used to add randomness to optimal solution to get minimal set ready for merging. These steps are executed until we get optimal solution for our problem.

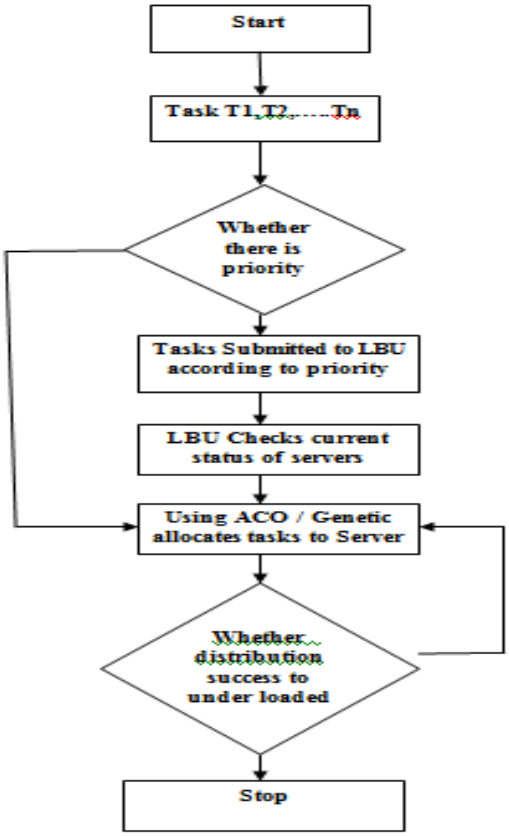


**Fig 2 Flow Chart.**

In above figure we can see that the flow of our proposed system when user submits its request in the form of cloudlets to LBU then LBU checks the current status using updated pheromone table information and also verifies whether its important task with assigned priority or normal tasks before allocating to appropriate server on basis of ACO or Genetic algorithm.

**4. Results and Discussion**

In this section of paper we describe how we used CloudSim environment for simulation of different load scenarios using ACO and Genetic optimization techniques.

```
Server:0has allocation cloudlet are:[1, 2, 3]

Server:1has allocation cloudlet are:[4]

Server:2has allocation cloudlet are:[5]

Server:3has allocation cloudlet are:[6]
New cloudlet request with following specification :
id:7
pesno:1
length:250000
fileSize:300
outputsize:300
EACH SERVER CAN HANDLE MAXIMUm 3 JOBS
```

**Fig 3 Status of servers**

As we mentioned earlier that LBU checks the status of each server before allocating new task to appropriate server, in fig 3 we can see that server 0 has been allocated cloudlets 1,2 & 3. While server 1, 2 and 3 has been given cloudlets 4,5 & 6 respectively for execution. Fig 3 also shows new cloudlet with id 7 just came to LBU.

```
Server with shortest distance of: 1596.5485295016715
Server with shortest distance of: 1596.5485295016713
Newly arrived task Waiting for LBU to finish random walk!
Found best so far: 1596.5485295016713
Underloaded shortest distance server is:1
Cloudlet 7 has allocated srever is::1
Simulation completed.
*****Datacenter: Datacenter_0*****
User id          Debt
3                142.4
************************************
CloudSimExample2 finished!
BUILD SUCCESSFUL (total time: 1 second)
```

**Fig 4 Task allocation using ACO**

In figure 4 we assigned incoming task to appropriate server using ACO technique to get under loaded server with shortest distance to transfer task from head node to destination node.

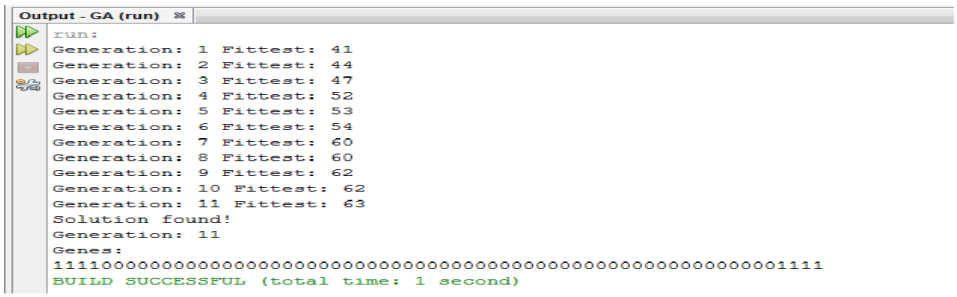


**Fig 5 Genetic optimization.**

In figure 5 we explained the result obtained after implementing Genetic technique in CloudSim environment. If we have large number of servers available for after applying genetic optimization we will get fittest server to transfer task from head node to destination node.

In this section we compared ACO and Genetic optimization technique to get best solution on basis of parameters such as Execution time and Cost.

**Table 1**

| Cloudlet Id | Execution Time using ACO (ms) | Execution Time using Genetic Optimization (ms) |
|---|---|---|
| 1 | 2800 | 2864 |
| 2 | 2800 | 2864 |

| | | |
|---|---|---|
| 3 | 3000.09 | 3064.09 |
| 4 | 3800.01 | 3864.01 |
| 5 | 5600 | 5664.08 |

Table 1 shows comparison of ACO and Genetic by considering execution time parameter. After comparison we found that ACO algorithm much better than Genetic optimization for execution parameter.

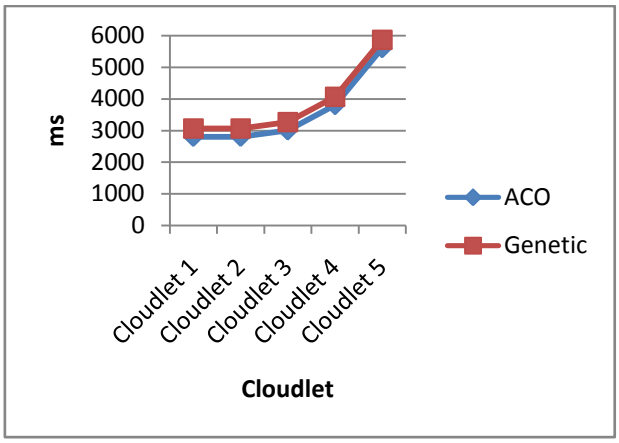


**Fig 6 Performance analysis of ACO with GA**

In figure 6 we can analyze performance of ACO and GA on basis of execution time parameter and got results from which we draw graph, from this result we can see that ACO is slightly better.

**Table 2**

| Cloudlet Id | Throughput ACO (per sec) | Throughput of Genetic Optimization (per sec) |
|---|---|---|
| 1 | 0.35 | 0.34 |
| 2 | 0.35 | 0.34 |
| 3 | 0.33 | 0.32 |
| 4 | 0.26 | 0.25 |
| 5 | 0.17 | 0.16 |

Table 2 showing throughput specifications obtained after ACO and GA implementation in CloudSim environment.

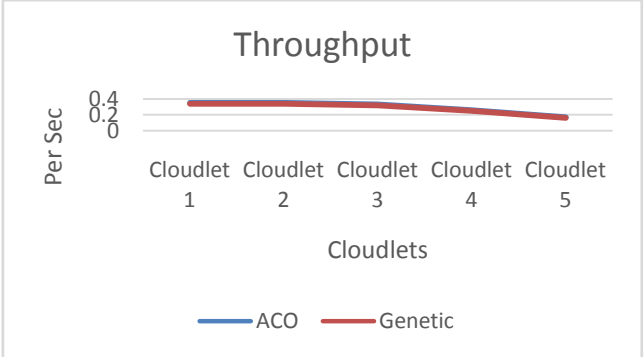


**Fig 7 Performance analysis of ACO with GA.**

From figure 7 we can conclude that throughput decreases due to increase in number of cloudlets assigned to servers.

**5. Conclusion**

In this approach, we try to resolve task load problem by analyzing performance of ACO and Genetic optimization Algorithm. We can give QoS to end user by using ACO algorithm rather than using Genetic technique. We successfully analyzed these load balancing approaches based on Cost, Throughput and execution time. We are able to provide higher preference to important task and provide service immediately.

**References:**

1. Ratan mishra, Anant jaiswal, (2012). Ant colony optimization solution for load balancing. IJWest, pages 33-50.
2. Kousik Dasgupta, Brototi Manda, (2013). A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing. International Conference on Computational Intelligence 10, pages 340-347.
3. Nader Mohamed, JameelaAl-Jaroodi , AbdullaEid,, (2013). A dual-direction technique for fast file downloads with dynamic load balancing in the Cloud. Journal of Network and Computer Applications 36, pages 1116-1130.
4. Zeng Zeng, Bharadwaj Veeravalli, (2014). Optimal metadata replications and request balancing strategy on cloud data centers. J. Parallel Distrib. Comput. 74, Pages 2934–2940.
5. Nikos D. Lagaros, (2014). An efficient dynamic load balancing algorithm. Comput Mech, pages 59–76.

**Corresponding Author Full details:**

Senthil Murugan Balakrishnan is a senior assistant professor of Information Technology and Engineering at VIT University, India. He graduated in Computer Applications in 2007 and pursuing his Ph.D. at VIT University. His research interests include parallel and distributed computing, SOA, network programming and Future Internet. He has received grants for deploying the satellite image processing application in the computational grid. He began his research career in developing the resource broker for the private cloud infrastructure present in VIT and further contributions include design and implementation of middleware for Future Internet models.