2012 14th International Conference on Modelling and Simulation

Load Balancing of Nodes in Cloud Using Ant Colony Optimization

Kumar Nishant, Pratik Sharma, Vishal Krishna,
Chhavi Gupta and Kuwar Pratap Singh
Department of CSE and ICT,
Jaypee University of Information Technology,
Solan-173234, Himachal Pradesh, India

Nitin and Ravi Rastogi
Department of CSE and ICT,
Jaypee University of Information Technology,
Solan-173234, Himachal Pradesh, India

***Abstract*—In this paper, we proposed an algorithm for load distribution of workloads among nodes of a cloud by the use of Ant Colony Optimization (ACO). This is a modified approach of ant colony optimization that has been applied from the perspective of cloud or grid network systems with the main aim of load balancing of nodes. This modified algorithm has an edge over the original approach in which each ant build their own individual result set and it is later on built into a complete solution. However, in our approach the ants continuously update a single result set rather than updating their own result set. Further, as we know that a cloud is the collection of many nodes, which can support various types of application that is used by the clients on a basis of pay per use. Therefore, the system, which is incurring a cost for the user should function smoothly and should have algorithms that can continue the proper system functioning even at peak usage hours.**

***Keywords*-Ant colony optimization; Cloud computing; Grid networks; Load balancing.**

I. Introduction

With the increasing popularity of cloud computing, the amount of processing that is being done in the clouds is surging drastically. A cloud is constituted by various nodes which perform computation according to the requests of the clients. As the requests of the clients can be random to the nodes they can vary in quantity and thus the load on each node can also vary. Therefore, every node in a cloud can be unevenly loaded of tasks according to the amount of work requested by the clients. This phenomenon can drastically reduce the working efficiency of the cloud as some nodes which are overloaded will have a higher task completion time compared to the corresponding time taken on an under loaded node in the same cloud. This problem is not only confined only to cloud but is related with every large network like a grid, etc. In this paper, we propose an efficient algorithm, based on ACO for better distribution of workload among the nodes of a cloud.

ACO has been previously used by researchers in grid computing for addressing load balancing [11], job scheduling [10] and related problems. Therefore, efficient use of this algorithm will contribute to better working of each of these large networks and help in the better utilization of resources available. As availability of resources is a major hindrance for the growth of cloud in countries like India, these algorithms would help in better utilization of available resources and expanding cloud at a much lower cost by saving the resources cost at each node. These types of algorithm can also be utilized by large organizations who ask users to donate extra CPU cycles for their own processing [18].

The rest of the paper is as follows: Section II provides the survey on the previous algorithms. Section III and IV explains the concepts of cloud computing and ant colony optimization. Section V and VI explains the proposed algorithm and pheromone updation. Section VII explains the pseudocode of the proposed algorithm. Section VIII explains the difference between the existing and proposed algorithms followed by the conclusion and references. We have also provided the pseudocode of our proposed algorithm.

II. Survey-Previous Algorithms

In previous works, the researchers have used the concept of ACO to build upon a Load balancing solution set within nodes of a cloud system [1]. The solution set was continuously updated by the ants pheromone trails, which move in iteration in the system. In this work the ants uses the basic pheromone updating formula and node selection formula of the ACO to distribute evenly the work loads of nodes in a cloud.

For efficient load balancing of work in cloud, tier-wise distribution of nodes is also suggested [4], in this the nodes are distributed in three tier structure such that the work is properly distributed among the nodes. In this hierarchy the 1st level (Top-level) nodes are used for the proper distribution of work among the nodes of 2nd level. Simultaneously the 2nd level distributes the work logically among the 3rd level nodes, which in turn-process their part of work. Thus, this system ensures the proper distribution of load among all levels.

The researchers around the globe have been working on the concept of load balancing of nodes in a networked architecture like a cloud, grid, etc. using the principles of Ant Colony Optimization. In one of these studies [5], an algorithm for node balancing suggested that the ants procreate at node level when need arises, and simultaneously

978-0-7695-4682-7/12 $26.00 © 2012 IEEE
DOI 10.1109/UKSim.2012.11

Figure 1. A Cloud

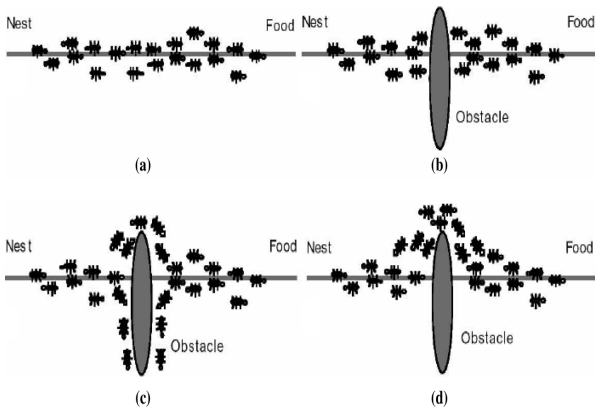

Figure 2. Modification in ants path upon encountering an obstacle

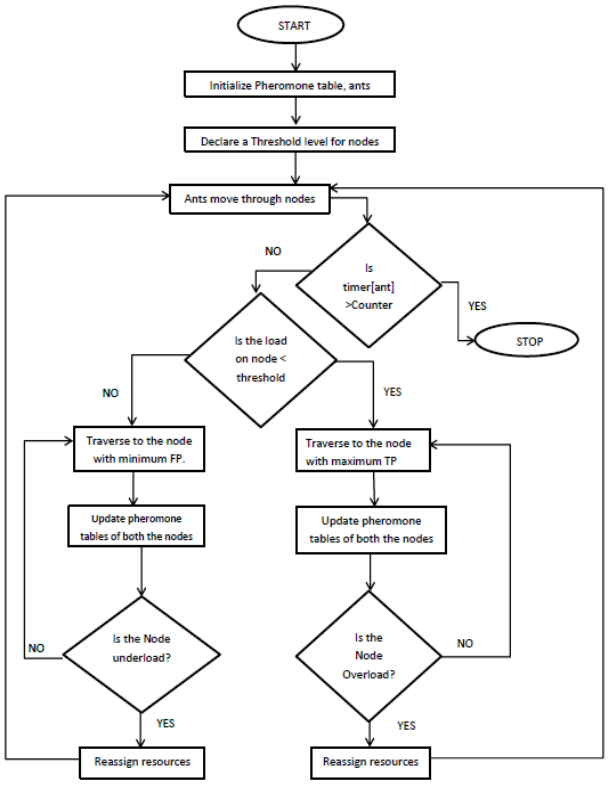

Figure 3. Flowchart

they commit suicide when their task is completed. In this algorithm the ant care for every node they visit and record their data for future decision making. In an another application of ant colony in grid computing a balanced approach of job scheduling called BACO was suggested [15]. The work aimed at reducing the computation time of each job execution in a grid network by considering the resource status and the size of the given job, the algorithm chooses optimal resources to process the submitted jobs by applying the local and global pheromone update technique to balance the system load.

In addition to this, in a relevant work, the concepts of ACO were applied in a more practical way [17] where the pheromone values were updated by adding a prize, punishment and load balancing coefficient to the original values. The value of pheromones were updated after job completion in such a way that if job was completed successfully, the pheromone values were updated by the prize coefficients whereas if vice-versa they were updated by the punishment coefficient. In all these previous algorithms there have been many shortcomings in application of ACO, like synchronizing ant movements for optimum data balancing, as properly moving ants would provide an efficient and better solution. We would discuss such algorithm in following sections.

III. Cloud Computing

Cloud computing is based on several other computing research areas such as HPC, virtualization, utility computing and grid computing. The improved computer hardware's and internet bandwidth has contributed immensely to the growth of cloud computing in today's world. Broadly the cloud is treated as service oriented approach and is classified among three popular types of service: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) [12,13]. Currently in cloud computing many nodes come together to provide the services to the users.

The fig. 1 above gives a brief idea about the cloud In the cloud system, the users can access the applications

faster with a good internet connection. There are many cloud computing service providers (CCSP) like-azure, etc. As these CCSP host the contents in form of applications on their nodes they charge their customers on the basis of data utilization. So, main drawback of cloud lies in the fact that if the service providers gets slow due to traffic overload or any other factors the customers would be more inclined to switch to other service providers. Therefore, efficiency is a major factor for the performance of cloud and an area of focus for these CCSP. One of the most popular application on cloud in wireless area is Apple's App store [14] which is creative and as well as popular and it started a culture of applications on software, and has been followed up by Microsoft, Google, etc.

IV. Ant Colony Optimization

ACO is inspired from the ant colonies that work together in foraging behavior. In fact the real ants have inspired many researchers for their work [6,8], and the ants approach has been used by many researchers for problem solving in various areas. This approach is called on the name of its inspiration ACO. The ants work together in search of new sources of food and simultaneously use the existing food sources to shift the food back to the nest.

The ethologists were jeopardized for many years as they wondered how even a blind ant was able to follow its fellow ants and exactly reached the food sources. They found that the ants leave a pheromone trail upon moving from one node to another. By following the pheromone trails, the ant subsequently came to the food sources. The intensity of the pheromone can vary on various factors like the quality of food sources, distance of the food, etc. The ants use these pheromone trails to select the next node. The ants can even modify their paths upon encountering any obstacles in their path. Figure 2, gives a brief idea about this scenario. This phenomenon of the ants was used in many algorithms for optimization where the ants follow each other through a network of pheromone paths. The ants upon traversal from one node to another update the pheromone trail of that path, so a path becomes more feasible if more ants traverse upon it. Paths that have the highest pheromone intensity have the shortest distance between the point and the best food source. The movements of these ants independently update a solution set.

The Traversal of ants in this system is generally of two types:

1) Forward movements-In this type of movement the ants move for extracting the food, or searching for the food sources.
2) Backward movements-In this type of movements the ants after picking up food from the food sources traverse back to the nest for storing their food.

The ACO is a unique algorithm for some of the reasons like the optimum solution is built not by a single entity but various entities, which traverse the length and breadth of the network and then these individually build upon a solution. Many researchers [9] to improve upon the results have also improvised upon the pheromone updating phenomenon of the ACO. It has been used by the researchers to improve upon various tasks such as task scheduling or optimizations in satellite networks [7].

V. Proposed Algorithm

In our paper, the ACO is used for load balancing. The approach aims at efficiently distribution of the load among the nodes and such that the ants never encounter a dead end for movements to nodes for building an optimum solution set. In our algorithm, first a Regional load balancing node (RLBN) [3] is chosen in a CCSP, which will act as a head node. We would be referring to the RLBN as head node in the rest of the paper. The selection of head node is not a permanent thing but a new head node can be elected if the previous node stops functioning properly due to some inevitable circumstances. The head node is chosen in such way that it has the most number of neighboring nodes, as this can help our ants to traverse in most possible directions of the network OF CCSP.

The ants in our proposed algorithm will continuously originate from the Head node. These ants traverse the width and length of the network in such a way that they know about the location of underloaded or overloaded nodes in the network. These Ants along with their traversal will be updating a pheromone table, which will keep a tab on the resources utilization by each node. We also proposed the movement of ants in two ways similar to the classical ACO, which are as follows:

1) Forward movement-The ants continuously move in the forward direction in the cloud encountering overloaded node or under loaded node.
2) Backward movement-If an ant encounters an overloaded node in its movement when it has previously encountered an under loaded node then it will go backward to the under loaded node to check if the node is still under loaded or not and if it finds it still under loaded then it will redistribute the work to the under loaded node. The vice-versa is also feasible and possible.

The main task of ants in the algorithm is to redistribute work among the nodes. The ants traverse the cloud network, selecting nodes for their next step through the classical formula given below, where the probability P_k of an ant, which is currently on node r selecting the neighboring node s for traversal, is:

$$P_k(r,s) = \frac{[\tau(r,s)][\eta(r,s)]^\beta}{[\tau(r,u)][\eta(r,u)]^\beta} \quad (1)$$

where,

r = Current node,

s = Next node,

τ = Pheromone concentration of the edge,

η = The desirability of the move for the ant (if the move is from an under loaded node to overloaded node or vice-versa the move will be highly desirable),

β = Depends upon the relevance of the pheromone concentration with the move distance.

However, with continuously originating ants at an interval of $\triangle t$, the overload incurred by network would increase as the number of paths followed by the ants would increase so would the cost for their maintenance and thus the network performance would take a beating. Therefore, we would keep their numbers in a limit [15]. We can keep their numbers in a limit by setting a suicide timer on the ant, which when reaches zero the ant will terminate itself. The selection of timer value would depend on the size and number of nodes in the network. The overload would depend too much on the interval $\triangle t$, the smaller the overload larger the overhead and vice-versa. However, higher the number of ants more frequent would be the data changes and load balancing and thus network efficiency. For this reason, if we could limit the number of ants in the network for a good trade-off between the need to keep collecting fresh data and reduce variance, and the need to avoid congestion of the ants as well.

VI. Pheromone Updation

The ant will use two types of pheromone for its movement. The type of pheromone being updated by the ant would signify the type of movements of the ant and would tell about the kind of node the ant is searching for. The two types of pheromones updated by the ants are as follows:

1) Foraging Pheromone (FP)-In a typical ACO the ant uses foraging pheromones to explore new food sources. In our algorithm the ant would lay down foraging pheromone after encountering under loaded nodes for searching overloaded nodes. Therefore, after an ant comes up to an under loaded node it will try to find the next path through foraging pheromone. The formula for updating this pheromone would be:

$$FP(t+1) = (1-\beta_{eva})FP(t) + \sum_{k=1}^{n} \Delta FP \quad (2)$$

where,
β_{eva} = Pheromone evaporation rate,
FP = Foraging pheromone of the edge before the move,
$FP(t+1)$ = Foraging pheromone of the edge after the move,
$\triangle FP$ = Change in the FP.

2) Trailing Pheromone (TP)-In a typical ACO the ant uses trailing pheromone to discover its path back to the nest.
However, in our algorithm the ants would use this to find its path to the under loaded node after encountering overloaded node. Therefore, after an ant encounters an overloaded node it will try to trace back the under loaded node through the trailing pheromone. The formula for updating this pheromone would be:

$$TP(t+1) = (1-\beta_{eva})TP(t) + \sum_{k=1}^{n} \Delta TP \quad (3)$$

where,
β_{eva} = Pheromone evaporation rate,
TP = Tracing pheromone of the edge before the move,
$TP(t+1)$ = Tracing pheromone of the edge after the move,
$\triangle TP$ = Change in the TP.

Therefore, the ants use these trails according to the kind of nodes they encounter.

The main aim of the two types of pheromone updation is to classify the ants according to the types of nodes they are currently searching for. The ants after originating from the head node, by default follow the Foraging pheromone, and in the process, they update the FP trails according to the formula. After coming upon an overloaded node they follow the Trailing Pheromones and simultaneously update the TP trails of the path. After reaching an underloaded node of the same type they update the data structure so as to move a particular amount of data from the overloaded node to under loaded node. Ants then select a random neighbor of this node, and if they encounter an underloaded node they start following the FP to trace an overloaded node, therefore they repeat the same set of tasks repeatedly in a network to improve the network performance.

While following on the TP upon encountering an underloaded node the ants will store information about the node in list which would include data like utilization ratio, free space and current tasks which can be used by the system to configure the best overloaded nodes suitable whose tasks could be relocated to these nodes and these will be decided by the factors like distance between the two nodes in question and the tasks which have to be relocated thus influencing the decision of load balancing. The tasks, which will be relocated, will be decided according to the already existing tasks at the underloaded node so that there be no clashes of interests. After each successful relocation of data between nodes the ants timer would be checked and if zero the particular ant would be terminated.

VII. Pseudocode

```
for(i=0; i<totalnodes; i++)
{
for(j=0; j<totalnodes; j++)
{
if(i&j == neighbouring nodes)
Pheromone table[][] = assign value;
}}

destinationnode=random (from_neighbouring_nodes
_of_headnode)
antoriginates(headnode)
antmoves(headnode, destinode)
if(load[destinode]>=threshold) {
do{
min=999999;
diff=0;
for(i=0; i<all neighbouring nodes of current node; i++) {
if((load[i]<threshold)&&(diff¡threshold-load[i])) {
nextnode=i;
flag=1;
diff=threshold-load[i]; } }

if(flag==0) {
for(i=0; i<all neighbouring nodes of current node; i++) {
if(tp[i]<min) {
min=tp[i];
nextnode=i;} } }
antmoves(currentnode, nextnode);
tpupdate(curentnode, nextnode);
currentnode=nextnode;
}while(load[currentnode]<threshold)
redistributeload(destinode, currentnode);
}
else
{
do {
max=0;
diff=0;
for(i=0; i<all neighbouring nodes of current node; i++) {
if((load[i]>threshold)&&(diff<threshold-load[i]))

{
nextnode=i;
flag=1;
diff=threshold-load[i];}}
if(flag==0) {
for(i=0;i<all neighbouring nodes of current node;i++) {
if(fp[i]>max) {
max=fp[i];
nextnode=i;}}}
fpupdate(curentnode,nextnode);
currentnode = nextnode;
}while(load[currentnode]<threshold)
redistributeload(destinode, currentnode);
}
```

VIII. Comparison

Table I
Table under Classical Algorithm

Nodes	A	B	C	D
A(O)	-	L	L	H
B(O)	L	-	L	H
C(M)	L	L	-	H
D(U)	H	H	H	-

Table II
Table under Modified Algorithm

Nodes	T.P. (O→U)				F.P. (U→O)			
	A	B	C	D	A	B	C	D
A(O)	-	L	M	H	-	L	L	L
B(O)	L	-	M	H	L	-	L	L
C(M)	L	L	-	H	M	M	-	L
D(U)	L	L	L	-	H	H	M	-

The Tables (1) and (2) shown above describe a situation of a cloud of nodes with varying loads. Their most frequent load type is denoted in the bracket (i.e. Overloaded-0, Medium loaded-M, Under loaded-U) and simultaneously the calculated pheromone table which indicates the level of particular pheromone type (i.e. High-H, Low-L, Medium-M) between corresponding nodes. Comparison of both these algorithms in normal situation would give similar results but our algorithm is modified for special situations which would ensure smooth functioning of the cloud. For example in a situation where the load type of node B is reversed that is the node becomes under loaded from its standard overloaded state, in the classical model the pheromone trails of node B is strong only with D which would make the most probable next state after B as D even in this reversed situation.

However, in our approach when a node is changed from an overloaded to an under loaded node the ants start to follow Foraging Pheromone instead of Trailing Pheromone and they simultaneously start to follow Trailing Pheromone instead of Foraging Pheromone when the node is changed to an overloaded state from an under loaded state. By, following this approach in this case the most likely state after B is not D and could be from any of the 3 states(refer to the table).

Therefore, in general it could be stated about the standard algorithm that after reversal of node types the ants traversing from now under loaded nodes would traverse to then under loaded nodes only which still could be under loaded and similarly for the overloaded nodes. But they are required to traverse to the opposite kind of nodes in both these cases. This task can be achieved from our modified approach which would make sure that the ants traversing from now under loaded node make use of the F.P. which would be

unbiased and would provide a greater chance of traversing to a overloaded node than compared to the previous approach. This aspect is the most important feature of this modified algorithm, and could be utilized in the situations in which the type of load on nodes can vary and thus not affecting the efficiency of ants with which they chose the opposite types of nodes.

IX. Conclusion

This is a modified approach of ant colony optimization that has been applied from the perspective of cloud or grid network systems with the main aim of load balancing of nodes. The main benefit of this approach lies in its detections of overloaded and underloaded nodes and thereby performing operations based on the identified nodes. This simplistic approach elegantly performs our task of identification of nodes by the ants and tracing its path consequently in search of different types of nodes. We have used the same concepts of Ant colony optimizations and have only modified the concepts where forward and trailing pheromones are used according to our convenience.

The way in which ants are created and their functionalities according to the pheromones trails and node encountered is clearly visible from the segment of pseudocode provided in the appendix. This modified algorithm has an edge over the original approach in which each ant build their own individual result set and it is later on built into a complete solution.

However, in our approach the ants continuously update a single result set rather than updating their own result set. In this way, the solution set is gradually built on and continuously improved upon rather than being compiled only once in a while. The other advantage of the approach lies in the fact that the task of each ant is specialized rather than being general and the task depends on the type of first node that was encountered whether it was overloaded or underloaded.

References

[1] S. Banerjee, I. Mukherjee and P.K. Mahanti, Cloud Computing Initiative using Modified Ant Colony Framework, World Academy of Science and Technology, 56, pp. 221-224, 2009.

[2] Y. Li, A Bio-inspired Adaptive Job Scheduling Mechanism on a Computational Grid, International Journal of Computer Science and Network Security, 6(3B), pp. 1-7, 2006.

[3] Z. Zhang and X. Zhang, A Load Balancing Mechanism Based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation, Proceedings of the 2nd International Conference on Industrial Mechatronics and Automation, pp. 240-243, 2010.

[4] S.C. Wang, K.Q. Yan, W.P. Liao and S.S. Wang, Towards a Load Balancing in a Three-level Cloud Computing Network, Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology, pp. 108-113, 2010.

[5] M. Salehi and H. Deldari, Grid Load Balancing using an Echo System of Intelligent Ants, Proceedings of the 24th IASTED International Conference on Parallel and Distributed Computing and Networks, pp. 47-52, 2006.

[6] M. Dorigo, V. Maniezzo and A. Colorni, Ant System: Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man, and Cybernetics, PP. 29-41, 1996.

[7] C.W. Chiang, Y.C. Lee, C.N. Lee and T.Y. Chou, Ant Colony Optimization for Task Matching and Scheduling, IEE Proceedings on Computers and Digital Techniques, 153 (6), pp. 373-380, 2006.

[8] M. Dorigo, M. Birattari and T. Stutzle, Ant Colony Optimization-Artificial Ants as a Computational Intelligence Technique, IEEE Computational Intelligence Magazine, pp. 1-12. 2006.

[9] J. Sun, S. Xiong and F.M. Guo, A New Pheromone Updating Strategy In Ant Colony Optimization, Proceedings of the International Conference on Machine Learning and Cybernetics, pp. 620-625, 2004.

[10] K.u Ruhana, K. Mahamud, H. Jamal and A. Nasir, Ant Colony Algorithm for Job Scheduling in Grid Computing, Proceedings of the Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, pp. 40-45, 2010.

[11] H. Jamal, A. Nasir, K. Ruhana, K. Mahamud and A.M. Din, Load Balancing Using Enhanced Ant Algorithm in Grid Computing, Proceedings of the Second International Conference on Computational Intelligence, Modelling and Simulation, pp. 160-165, 2010.

[12] C. Gong, J. Liu, Q. Zhang, H. Chen and Z. Gong, The Characteristics of Cloud Computing, Proceedings of the 39th International Conference on Parallel Processing Workshops, pp. 275-279, 2010.

[13] http://searchcloudcomputing.techtarget.com

[14] http://www.techsmith.com/morae/whitepaper/ux20.asp

[15] R.A. Arnous, H.A. Arafat and M.M. Salem, Improving the Load Balancing within the Data Network via Modified AntNet Algorithm, Proceedings of the 5th International Conference on Information and Communication Technology, pp. 189-195, 2007.

[16] R. Chang, J. Chang and P. Lin, Balanced Job Assignment Based on Ant Algorithm for Grid Computing, Proceedings of the IEEE Asia-Pacific Services Computing Conference, pp. 291-295, 2007.

[17] H. Yan, X. Shen, X. Li and M. Wu, An Improved Ant Algorithm for Job Scheduling in Grid Computing, Proceedings of the International Conference on Machine Learning and Cybernatics, 5, pp. 2957-2961, 2005.

[18] http://boinc.berkeley.edu/