IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

Improving Efficiency by Balancing the Load Using Enhanced Ant Colony Optimization Algorithm in Cloud Environment

Ashwini L[1], Nivedha G[2], Mrs A.Chitra[3]

[1, 2]Student, Kingston Engineering College

[3]Assistant Professor, Department of Computer Science and Engineering, Kingston Engineering College

Abstract
As Cloud Computing is growing rapidly and clients are demanding more services and better results, load balancing for the Cloud has become a very interesting and important research area. Many algorithms were suggested to provide efficient mechanisms for assigning the client's requests to available Cloud nodes. These approaches aim to enhance the overall performance of the Cloud and provide the user more satisfying and efficient services. 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 balancing the 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. 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.*

1. Introduction

1.1 Cloud Computing

"Cloud computing" is a term, which involves virtualization, distributed computing, networking, software and web services. A cloud consists of several elements such as clients, datacenter and distributed servers. It includes fault tolerance, high availability, scalability, and flexibility, reduced overhead for users, reduced cost of ownership, on demand services [2].

In Cloud computing services can be used from diverse and widespread resources, rather than remote servers or local machines. According to the

NIST "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, server, storage, application, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1].

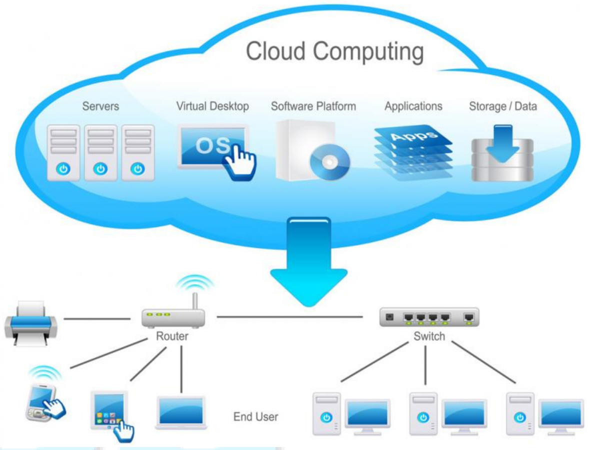

Figure 1: Cloud Computing

1.2 Types of Cloud Computing

Private cloud – The cloud infrastructure operated solely for a single organization.
Public cloud – Services are rendered over a network that is open for public use.
Community cloud- Shares infrastructure between several organizations from a community.
Hybrid cloud – Composition of two or more clouds with unique entities.
Distributed cloud – Provided by a set of distributed machines that are running at different location [1].

1.3 Service models

Infrastructure as a Service (IaaS): The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications [2].
Platform as a Service (PaaS): The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider [2].

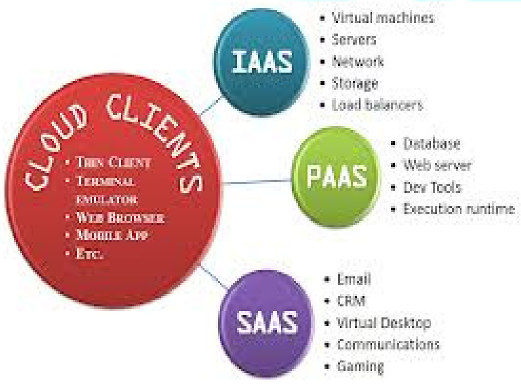

Figure 2: Cloud Computing Architecture

Software as a Service (SaaS): The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure2. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface.

1.4 Issues in cloud computing

Security, data, Load Balancing, Performance, etc are the issues in cloud computing [2].

2. Load Balancing

Load balancing is one of the central issues in cloud computing. The load can be CPU load, memory capacity, delay or network load. Load balancing is the process of distributing the load among various nodes of a distributed system to improve both resource utilization and job response time while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or doing very little work. The load can be balanced either in hardware or software side.

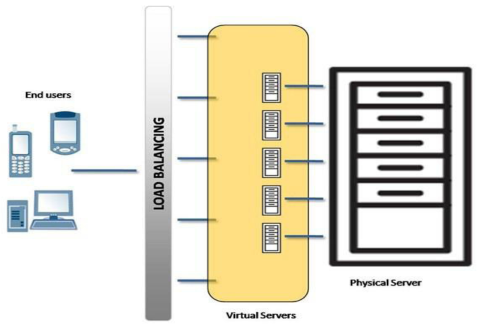

Figure 3: Hardware side load balancing.

2.1 Need of Load Balancing in Cloud Computing

Load balancing in clouds is a mechanism that distributes the excess dynamic local workload evenly across all the nodes. It is used to achieve a high user satisfaction and resource utilization ratio, making sure that no single node is overwhelmed, hence improving the overall performance of the system. Proper load balancing can help in utilizing the available resources optimally, thereby minimizing the resource consumption. It also helps in implementing fail-over, enabling scalability, avoiding bottlenecks and over-provisioning, reducing response time etc [2].

2.2 Challenges in Load balancing

Overhead Associated - determines the amount of overhead involved while implementing a load balancing algorithm. It is composed of overhead due to movement of tasks, inter processor and inter-process communication. This should be minimized so that a load balancing technique can work efficiently [3].
Throughput - is used to calculate the no. of tasks whose execution has been completed.
Performance – is used to check the efficiency of the system. It has to be improved at a reasonable cost e.g. reduce response time while keeping acceptable delays.
Resource Utilization - is used to check the utilization of resources. It should be optimized for an efficient load balancing.
Scalability - is the ability of an algorithm to perform load balancing for a system with any finite number of nodes. This metric should be improved.
Response Time - is the amount of time taken to respond by a particular load balancing algorithm in a distributed system. This parameter should be minimized.

Fault Tolerance - is the ability of an algorithm to perform uniform load balancing in spite of arbitrary node or link failure. The load balancing should be a good fault-tolerant technique [3].

3. Survey on various load balancing algorithms

3.1 Round Robin Algorithm

Round Robin is a very famous load balancing algorithm, in which the processes are divided between all processors. The process allocation order is maintained locally independent of the allocations from remote processors. In Round Robin, it send the requests to the node with the least number of connections, so at any point of time some node may be heavily loaded and other remain idle [2], this problem is reduced by CLBDM.

3.2 Central Load Balancing Decision Model (CLBDM)

CLBDM is a central load balancing decision model, which is suggested by Radojevic and Mario Zagar [9], it's based on session switching at the application layer. The improvement is that, in the cloud it calculated the connection time between the client and the node, and if that connection time exceeds a threshold then connection will be terminated and task will be forwarded to another node using the regular Round Robin rules.

3.3 Map Reduce-based Entity Resolution

Map Reduce is a computing model and an associated implementation for processing and generating large datasets. Map task and reduce task two main task in this model which written by the user, Map takes an input pair and produces a set of intermediate value pair and Reduce task accepts an intermediate key and a set of values for that key and merges these values to form a smaller set of value. Map task read entities in parallel and process them, this will cause the Reduce task to be overloaded.

3.4 Load Balancing Min-Min Algorithm (LBMM)

Wang suggested an algorithm called LBMM. LBMM has a three level load balancing framework. In first level LBMM architecture is the request manager which is responsible for receiving the task and assigning it to service manager, when the service manager receives the request; it divides it into subtask and assigns the subtask to a service node based on node availability, remaining memory and the transmission rate which is responsible for execution the task [4].

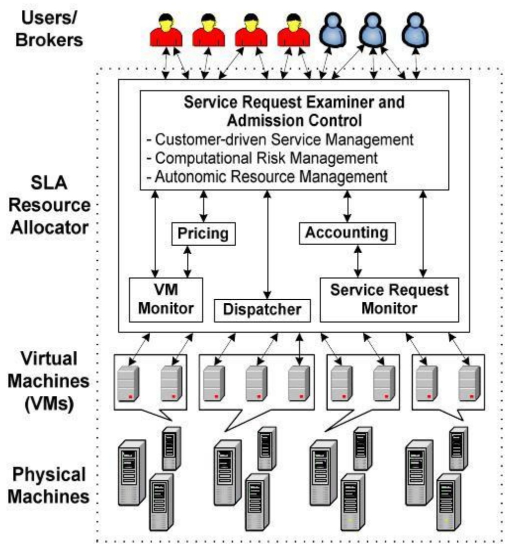
Users/ Brokers
SLA Resource Allocator
Service Request Examiner and Admission Control
- Customer-driven Service Management
- Computational Risk Management
- Autonomic Resource Management
Pricing
Accounting
VM Monitor
Dispatcher
Service Request Monitor
Virtual Machines (VMs)
Physical Machines

Figure 4: Software side load balancing.

3.5 Ant colony optimization (ACO)

Kumar Nishant suggested an algorithm of ant colony optimization. In ACO algorithm when the request in initiated the ant start its movement. Movement of ant is of two ways*: Forward Movement*: Forward Movement means the ant in continuously moving from one overloaded node to another node and check it is overloaded or under loaded ,if ant find an over loaded node it will continuously moving in the forward direction and check each nodes .*Backward Movement*: If an ant find an over loaded node the ant will use the back ward movement to get to the previous node, in the algorithm if ant finds the target node then ant will commit suicide, this algorithm reduced the unnecessary back ward movement, overcome heterogeneity, is excellent in fault tolerance.

Here we are going to concentrate on Ant Colony Optimization Algorithm for balancing the load in order to improve the efficiency.

4. Ant Colony Optimization

ACO is inspired from the ant colonies that work together in foraging behaviour. In fact the real ants have inspired many researchers for their work 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 ethnologists 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 [4]. 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. 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 [7].

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 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].

The limitation of ACO algorithm is every ant build their own individual result set and it is later on built into a complete solution.

5. Proposed algorithm

This approach aims at efficient 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) 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 neighbouring nodes, as this can help our ants to traverse in most possible directions of the network OF CCSP [7].

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 under loaded 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:

5.1 Forward movement

The ants continuously move in the forward direction in the cloud encountering overloaded node or under loaded node.

5.2 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.
However, with continuously originating ants at some interval, 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. 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 time, 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.

6. 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:

6.1 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.

6.2 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.

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. After coming upon an overloaded node they follow the Trailing Pheromones and simultaneously update the TP trails of the path. After reaching an under loaded 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 neighbour of this node, and if they encounter an under loaded 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 under loaded 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 under loaded 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.

7. 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 under loaded 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.

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 under loaded. The implementation of this paper will be done in future work.

References

[1] "NIST Cloud Computing References Architecture" Special publication 500_292, September 2011.

[2] Sukalyan Goswami and Ajanta De Sarkar, "A Comparative study of Load Balancing algorithms in computational Grid environment", 2013 fifth International Conference on Computational Intelligence, Modelling and simulation.

[3] Klaithem Al Nuaimi, Nadar Mohamed, Marium Al Nuaimi and Jameela Al – Jaroodi, "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms", 2012 IEEE Second symposium on Network Computing and Applications.

[4] Huankai Chen, Professor Frank Wang, Dr Na Helian and Gbola Akanmu, "User-Priority Guided Min-Min Scheduling Algorithm For Load Balancing in Cloud Computing".

[5] Andrzej Goscinski, Michael Brock, Future Generation Computer Systems"Towards Dynamic and Attribute based Publication, Discovery and Selection for Cloud Computing", 2010 Elsevier.

[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, IEEE Proceedings on Computers and Digital Techniques, 153 (6), pp. 373- 380, 2006.