Cloud computing Network Problem and Storage solutions Using Ant colony optimization

1st Dr.Madhurendra Kumar
Assistant Professor, CSE Department
Lingayas University,Fridabad
Email: madhurgmca@gmail.com

ABSTRACT

Cloud computing is the most network-centric distributed computing, parallel computing and grid computing Where successful transition to Cloud will depend on a excel network foundation .One of the fundamental issues in this network environment and storage I related to task scheduling is flexible access and use of resources in network technology where service provider connected through Internet. Cloud computing disregards the need of having a complete infrastructure of hardware and software to meet users requirements and applications. Cloud users to understand and design more of the network to which they are exposed .This paper aim to address most of this network problem and their possible solution using Ant Colony Optimization (ACO) algorithm.

Keyword: Introduction, Critical Network, cloudsim, Ant Colony Optimization (ACO), Implementation, Conclusion.

1. INTRODUCTION

Cloud computing is a model system in which resources and services are abstracted from the underlying infrastructure and provided on order and at scale in a multi-tenant environment. From a networking standpoint, each service model requires the cloud contributor to expose more or less of the network and provide more or fewer networking capabilities to cloud users. Every service model requires cloud users to understand and design more of the network to which they are exposed. Network infrastructure is most exposed in the IaaS model and least in the SaaS model. The essential technological difference between the operation models is derived from the networking relationship between the cloud client and the cloud service provider. In a private cloud, the client and service provider are within the same reliable network boundary. In a public cloud, they are on various networks. In a hybrid cloud, a secured connection may exist between the user's and provider's networks, or the user's network may pull out into the provider's cloud (or the reverse). Community cloud is the structure depends on the charter and structural design of the organizations operating the cloud. Every cloud is some combination of a service and deployment model. In any case of the type of cloud, however, one fact remains true: no network means no cloud.

If no networks, users cannot access their cloud services. Without networks, applications, data, and users cannot move among clouds. Without networks, the infrastructure components that must work together to create a cloud cannot. There is an escalating proliferation of phones and other mobile devices that are being used to access applications and data from clouds across many various kinds of networks. Until recently, "mobile" mainly referred to these devices and the networks that support them with cloud infrastructure, applications and servers also have become mobile. It move from one part of a cloud to another or even from one cloud to another. It is making the network aware of and communicates to not just users accessing the cloud, but also the applications and data in the cloud. Extending and interconnect clouds, enabling application and data and user mobility between clouds. Providing consistent quality of service around the entire network. Enforcing policies on devices, users, data, and applications regardless of position and making the policy Enforcement points themselves mobile. Providing a consistent policy infrastructure, centralized control, separation of duties, and the capability to deliver federated sign-on and policy enforcement across clouds. Creating self-service directory, orchestration, and automation tools to provision all IT resources in the fabric providing the group of metrics directly from the fabric for analysis and response. Working with the Open Stack group to create an open network-as-a-service (NaaS) capability for provisioning networking resources in open cloud environments.

2. CRITICAL NETWORK

Networking Scenario we have to change because the rise of cloud service is changing what is happening on the network. In new infrastructure everything is becoming virtualized, communications is becoming programmable, and servers and applications have mobility. New applications use as data-intensive analytical, parallel and

clustered processing, telemedicine, remote experts, and community cloud services.

For example, mobile services access to everything and virtual desktops. New traffic use as predominantly server-to-server traffic and location-independent endpoints on both sides of a service or transaction. What we need to do with and to data has not changed. Data still needs to travel among the computing and storage components of an application and user. Security still necessity be applied to help make sure that the right devices, user and systems have access to the right data at the exact time while protecting against attacks, intrusions, and leaks. Different kinds of data and traffic have various levels of importance and network resource needs that met the whole network with quality-of-service (QoS) capabilities.

3 CLOUDSIM

Simulation is a procedure where a program models the activities of the system (CPU, network etc.,) by calculating the interaction connecting its different entities using mathematical formulas, or actually capturing and playing back observations from a construction system. Cloudsim is a framework developed by the GRIDS laboratory of university of Melbourne which deals seamless modeling, simulation and experimenting on designing cloud computing infrastructures.

3.1 Cloudsim Charectistics

Cloudsim can be used to model data storage, host, service brokers, scheduling and allocation policies of a large scaled cloud platform. Hence, the researcher has used cloudsim to model datacenters, hosts, VMs for experimenting in simulated cloud environment. Cloud supports VM provisioning at two levels:

(i) At the host level: It is possible to specify how much of the whole processing power of each core will be allocated to each VM known as VM policy Allocation.
(ii) At the VM level: The VM assigns a fixed amount of the available processing power to the separate application services (task units) which are hosted within its execution engine known as VM Scheduling.

In this paper, the ACO algorithm will be used for allocation of incoming group works to VMs at the VM level (VM Scheduling). All the VMs in a data center not necessary have a fixed amount of processing power but, it can vary with different computing nodes, and then to these VMs of different processing powers, the tasks/ requests (application services) are assigned or allocated to the most powerful VM and then to the lowest and so on. Hence, the performance parameter such as overall make span time is optimized (increasing resource utilization ratio) and the cost will be decreased.

4. CLOUD SCHEDULING BASED ACO

ACO is to put on the foraging behavior of ant colonies. When an ants group tries to search for the food, they use a special kind of chemical to communicate with each other. That chemical is referred to as pheromone. Initially, ants starts search their foods randomly. Once the ants find a path to food source, they leave pheromone on the path. An ants follow the concept to the food source by knowing pheromone on the ground. As this process continues, most of the ants attract to choose the shortest path as there have been a vast amount of pheromones accumulated on the root. The advantages of the algorithm are the use of the positive response, inner parallelism and extensible. The inaction phenomenon, or searching for to a certain extent, all alone found the same result exactly, can't further search for the solution space, making the algorithm converge to local optimal solution. It is clear that an ACO algorithm can be applied to any combinatorial problem as far as it is possible to define:

(i) Problem representation which allows ants to incrementally build/ modify solutions.
(ii)The heuristic desirability η of edges.
(iii) A constraint satisfaction method which forces the construction of feasible solutions.
(iv)A pheromone updating rule which specifies how to modify pheromone trail τ on the edges of the graph.A probabilistic transition rule of the heuristic desirability and of pheromone trail.

In this section, cloud task scheduling based ACO algorithm will be proposed. Decreasing the makespan of tasks is the basic ideas from the proposed method.

4.1.Problem Representation:

For solution it is represented as a graph G= (N, E) where the set of nodes N represents the VMs and tasks and the set of edges E the connections between the task and VM as shown in Figure. All ants are placed at the starting VMs randomly. During an iteration ants build solutions to the cloud scheduling problem by moving from one VM to another for next task until they complete a tour (all tasks have been allocated). Repetitions are indexed by t, 1< t< tmax, where tmax is the maximum number of iterations allowed

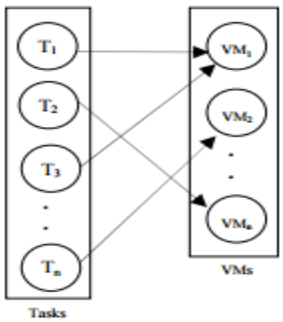

Figure 1. Problem representation of task scheduling based ACO.

4.2 Heuristic Desirability:

A very simple heuristic is used the inverse of expected execution time of the task i on VM j.
Constraint Satisfaction:
The constraint satisfaction method is implemented as a simple, short-term memory of the visited VM, in order to, avoid visiting a VM more than once in one ACO procedure and minimize time of the assigned couplings (task and VM).

4.3 Pheromone Updating Rule:

It is the one typical of ant system as shown in Equations 3, 4, 5, 6 and 7. Pheromone evaporates on all edges and new pheromone is deposited by all ants on visited edges; its value is proportional to the quality of the solution built by the ants.

4.4 Probabilistic Transition Rule:

The probabilistic transition rule, called random proportional, is the one typical of ant system as shown in Equation 1. The pseudo code of the proposed ACO algorithm and scheduling based ACO algorithm are shown in Algorithms 1 and 2 respectively. The main operations of the ACO procedure are initializing pheromone, choosing VM for next task and pheromone updating as following:

Algorithm: Scheduling based ACO algorithm

Input: Incoming Cloudlets and VMs List
Output: Print "scheduling completed and waiting for more Cloudlets"Steps:

1. Set Cloudlet List=null and temp_List_of_Cloudlet=null
2. Put any incoming Cloudlets in Cloudlet List in order of their arriving time
3. Do ACO_P while Cloudlet List not empty or there are more incoming Cloudlets

Set n= size of VMs list
If (size of Cloudlet List greater than n)
Transfer the first arrived n Cloudlets from Cloudlet List and put them on temp_List_of_Cloudlet
Else
Transfer all Cloudlets.from Cloudlet List and put them on temp_List_of_Cloudlet
End If
Execute ACO procedure with input temp_List_of_Cloudlet and n
End Do

4. **Print** "scheduling completed and waiting for *more Cloudlets"*
5. **Stop**

4. For k:=1 to m do

Compute the length Lk of the tour described by the k-th ant according to Equation 4.

Update the current_optimal_solution with the best founded solution.

5. Implementation and Experimental Results

We assume that tasks are mutually independent i.e., there is no precedence constraint between tasks and Tasks are not preemptive and they cannot be interrupted or moved to another processor during their execution.

5.1. Parameters Setting of Cloudsim

The experiments are implemented with 10 Datacenters with 50 VMs and 100- 1000 tasks under the simulation platform. The length of the task is from 1000 Million Instructions (MI) to 20000 MI. The parameters setting of cloud simulator are shown in Table 1

Table 1. Parameters setting of cloudsim

Entity Type	Parameters	Value
Task (cloudlet)	Length of Task	1000-20000
	Total Number of Task	100-1000
Virtual Machine	Total Number of VMs	50
	MIPS	500-2000
	VM Memory(RAM)	256-2048
	Bandwidth	500-1000
	Cloudlet Scheduler	Space_shared and Time_shared
	Number of PEs Requirement	1-4
Data Center	Number of Datacenter	10
	Number of Host	2-6
	VM Scheduler	Space_shared and Time_shared

5.2. ACO Parameters Evaluation and Setting

We implemented the ACO algorithm and investigated their relative strengths and weaknesses by experimentation. The parameters (α, β, p, tmax, m the number of ants and Q) considered here are those that affect directly or indirectly the computation of the algorithm. We tested several values for each parameter while all the others were held constant on 100 tasks. The default value of the parameters was α=1,β=1, ρ=0.5, Q=100, tmax=150 and m=8.

In each experiment only one of the values was changed, The values tested were:

α€ {0, 0.1, 0.2, 0.3, 0.4, 0.5}, β€ {0, 0.5, 1.5, 2,2.5, 3},

ρ€ {0, 0.1, 0.2, 0.3, 0.4, 0.5}, Q € {1, 100, 500,1000},

tmax € {50, 75, 100, 150} and *m € {1, 5, 8, 10,15, 20}*.

We also use the time in the cloudSim to record the makespan. The ACO performance for different values of parameters (α, β, p, tmax, m the number of ants and Q) has been evaluated. The ACO performance for different values of parameters (m: The number of ants, tmax, Q, , αand β) are shown from Figures 2 to 7. It can be seen that the best value of α is 0.3, the best value of β is 1, the best value of is 0.4, the best value of Q is 100, the best value of tmax is 150 and the best values of m is 10. In the following experiments we select the best value for α, β, , Q and m parameters but, the value 100 is selected for the tmax parameter to reduce the overhead of the ACO algorithm.

Table 2 shows the selected best parameters of ACO.

Figure 2. ACO performance for different values of ant numbers.

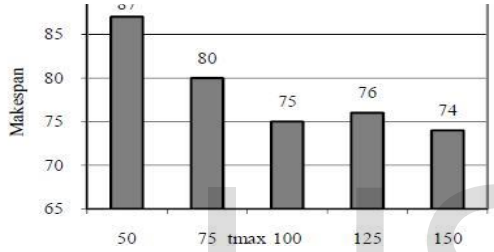

Figure 3. ACO performance for different values of tmax.

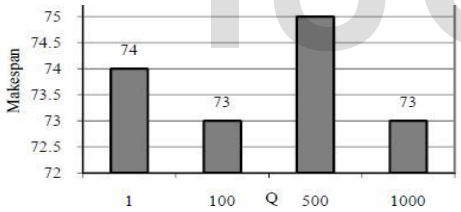

Figure 4. ACO performance for different values of Q.

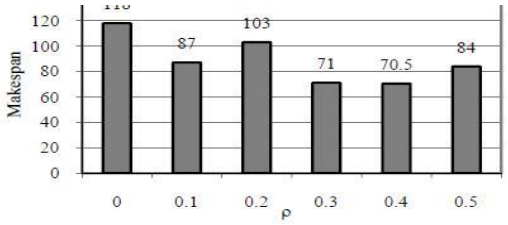

Figure 5. ACO performance for different values of RHO

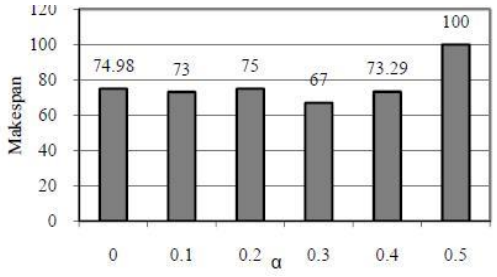

Figure 6. ACO performance for different values of alpha

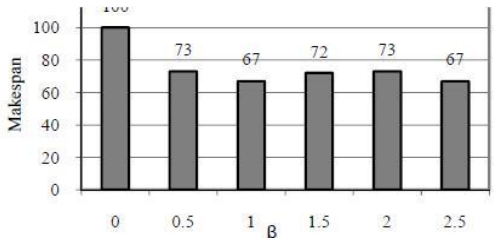

Figure 7. ACO performance for different values of beta.

Table 2. Selected parameters Of ACO.

Parameter	α	β	ρ	Q	m	tmax
Value	0. 3	1	0.4	100	10	100

5.3.Implementation Result of ACO,FCFS and RR

The following experiments, we compared the average make span with different tasks set. The average makes span of the ACO, RR and FCFS algorithms are shown in Figure 8. It can be seen that, with the increase of the quantity task, ACO takes the time less than RR and FCFS algorithms. This indicates that ACO algorithm is better than RR and FCFS algorithms.

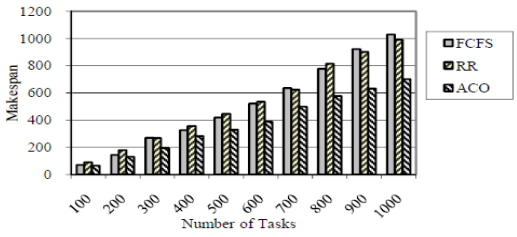

Figure 8. Average makespan of FCFS, RR and ACO

With help from statistics and probability theory, standard deviation (σ) shows how much deviation or dispersion exists from the average (mean), or expected value. A low standard deviation indicates that the data points tend to be very close to the mean; high standard deviation indicates that the data points are spread out over a large range of values (solving stagnation problem). Since, the standard deviation of never drops to zero, we are assured that the algorithm actively searches solutions which differ from the best-so-far found, which gives it the possibility of finding better

ones. Figure 9 shows the evolution of the standard deviation of the ACO over 10 runs.

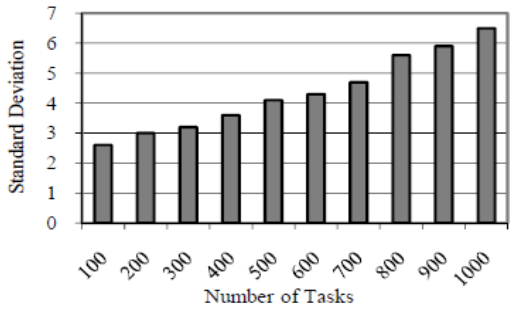

FIGURE 9. STANDARD DEVIATION OF ACO OVER 10 RUNS

The Degree of Imbalance (DI) measures the imbalance among VMs, which is computed by Equations 1 and 2.

$$Ti = \frac{TL\_Tasks}{Pe\_num_i * Pe\_mips} \quad (1)$$

Where, TL_Tasks is the total length of tasks which are submitted to the VMi

$$DI = \frac{Tmax - Tmin}{Tavg} \quad (2)$$

Where, Tmax, Tmin and Tavg are the maximum, minimum and average Ti respectively among all VMs. The average DI of each algorithm with the number of tasks varying from 100 to 1000 is shown in Figure 10. It can be seen that the ACO can achieve better system load balance than RR and FCFS algorithms.

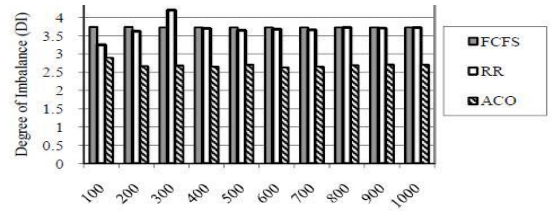

Figure 10. Average DI of FCFS, RR and ACO.

6.MY PROPOSED WORK

With help from multithreading java programming we can implement reduce the time complexity between consumer and cloud storage.
Now we'll illustrate a classic interaction between two threads: a storage and a Consumer. A storage thread creates messages and places them into a queue, while a consumer reads them out and displays them. To be realistic, we'll give the queue a maximum depth. And to make things really interesting, we'll have our consumer thread be lazy and run much slower than the storage. This means that stroge occasionally has to stop and wait for Consumer to catch up. The example below shows storage and consumer classes.

```
import java.util.Vector;

class Storage extends Thread
{
   static final int MAXQUEUE = 5;
   private Vector messages = new Vector();

   public void run()
           {
      try
      {
         while ( true )
            {
            putMessage();
            sleep( 1000 );
            }
       }
              Catch( InterruptedException e )
            {
            System.out.println(e);
            }
   }
private synchronized void putMessage()
      throws InterruptedException
        {

      while ( messages.size() == MAXQUEUE )
         wait();
      messages.addElement( new java.util.Date().toString()
);
      notify();
   }

   // Called by Consumer
   public synchronized String getMessage()
      throws InterruptedException
        {
      notify();
      while ( messages.size() == 0 )
         wait();
      String message = (String)messages.firstElement();
      messages.removeElement( message );
      return message;
   }
}

class Consumer extends Thread
{
  Storage producer;

   Consumer(Storage  p)
 {
      producer = p;
   }

   public void run()
{
      try
```

```
            {
        while ( true )
            {
            String message = producer.getMessage();
            System.out.println("Got message: " + message);
            sleep( 2000 );
        }
      }
      catch( InterruptedException e )
            {
            System.out.prinln(e);
            }
   }

   public static void main(String args[])
            {
     Storage producer = new Storage r();
      producer.start();
      new Consumer( producer ).start();
   }
}
```

7. CONCLUSION

Cloud computing is the most network based compute paradigm. A successful transition to Cloud will depend on a rock-solid network foundation that enables organizations to transition to the cloud at their own pace. We propose a cloud Ant colony optimization algorithm. It does this operation in order to perform efficient resource utilization and load balancing of the servers. The future enhancement of this suggested system could to modify the system performance by reducing the number servers present in the network using ACO algorithm for achieving cloud computing tasks scheduling has been presented. Firstly, the best values of parameters for ACO algorithm, experimentally determined. Then, the ACO algorithm in applications with the number of tasks varying from 100 to 1000 evaluated. Simulation results demonstrate that ACO algorithm outperforms FCFS and RR algorithms. In future work the effect of precedence between tasks and load balancing will be considered.

7.REFERENCES

[1]. Antony Rowstron and Peter Druschel, “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems”, IFIP/ACM International Conference on Distributed SystemsPlatforms.

[2]. Buyya R., Ranjan R., and Calheiros N., “Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities,” in Proceedings of the 7th High Performance Computing and Simulation Conference, Leipzig, Germany, pp. 1-11, 2009.

[3].Dorigo M. and Blum C., “Ant Colony Optimization Theory: A Survey,” in Theoretical Computer Science, vol. 344, no. 2, pp. 243-278, 2005.

[4].Dorigo M., Birattari M., and Stutzel T., “Ant Colony Optimization,” IEEE Computational Intelligence Magazine, vol. 1, no. 4, pp. 28-39, 2006.

[5].Fangzhe C., Ren J., and Viswanathan R., “Optimal Resource Allocation in Clouds,” in Proceedings of the 3rd International Conferenceon Cloud Computing, Florida, USA, pp. 418- 425, 2010.

[6].Gao K., Wang Q., and Xi L., “Reduct Algorithm Based Execution Times Prediction in Knowledge Discovery Cloud Computing Environment,” the International Arab Journal of Information Technology, vol. 11, no. 3, pp. 268- 275, 2014.

[7].Gao Y., Guan H., Qi Z., Hou Y., and Liu L., “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.

[8]. R. Buyya and M. Murshed. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing.Concurrency and Computation: Practice and Experience, 14(13-15), Wiley Press, Nov.-Dec., 2002.

[9]. Ghalem B., Tayeb F., and Zaoui W.,“Approaches to Improve the Resources Management in theSimulator Cloudsim,” in Proceedings of the Conference on Interactionand Confidence Building Measures in Asia,Lecture Notes in Computer Science, Istanbul,Turkey, pp. 189-196, 2010.

[10]. Hsu C. and Chen T., “Adaptive Scheduling Based on Quality of Service in HeterogeneousEnvironments,” in Proceedings of the IEEEInternational Conference on Multimedia and Ubiquitous Engineering, California, USA, pp. 1-6, 2010.

[11]. Ijaz S., Munir E., Anwar W., and Nasir W.,“Efficient Scheduling Strategy for Task Graphs in Heterogeneous Computing Environment,” theInternational Arab Journal of Information Technology, vol 10, no. 5, pp. 486-492, 2013.

[12].Kessaci Y., Melab N., and Talbi E., “A Pareto- Based GA for Scheduling HPC Applications on Distributed Cloud Infrastructures,” in Proceedings of the IEEE International Conference on High Performance Computing and Simulation, Istanbul, Turkey, pp. 456-462, 2011.

[13]. Lorpunmanee S., Sap M., Abdul A., andChompoo C., “An Ant Colony Optimization forDynamic Job Scheduling in Grid Environment,”in Proceedings of World Academy of Science, English and Technology, 2007.