PERFORMANCE ANALYSIS OF ANT BASED TASK SCHEDULING ALGORITHMS IN CLOUD ENVIRONMENT

D. Gupta1*, H. J. S. Sidhu2, G. Kumar3

1* CSE, Desh Bhagat University, Mandi Gobindgarh, Punjab, India
2 CSE, Desh Bhagat University, Mandi Gobindgarh, Punjab, India
3 IT, DAVIET, Jalandhar, Punjab, India

ABSTRACT:

Scheduling of tasks in cloud environment is a key research area. Cloud computing environment provides virtualized resources to applications dynamically. These resources are costly and often impossible for users to own them privately. Users are charged on a pay-per-use basis for these resources on cloud environment. The idea of task scheduling in cloud environment is to optimize the execution time so that the cost paid by the user is minimal. Ant based algorithms are very popular for task scheduling related problems in cloud computing. Ants destroy themselves at destination and do not return at source which makes them even more popular. This paper provides performance analysis of 2 ant based algorithms for task scheduling in cloud environment.

Keywords: ACO, VM, Data Centre, Cloud Computing, ETC.

[1] INTRODUCTION

Cloud computing is commercial implementation, which combines traditional computer and network technology such as distributed computing, parallel, computing, grid computing, utility computing, virtualization and network storage, etc. It is a kind of computing service mode, which is based on the Internet. The convenient access for shared
configurable computing resources (network, servers, storage, applications and services, etc.) and on-demand access for resources is realized [9].

The organization of paper is as following: Section 2 Introduction to the related work. Section 3 provides an overview to A2LB task scheduling algorithm. Section 4 provides an overview to ACO task scheduling algorithm Section 5 provides an insight into the results of A2LB & ACO scheduling algorithms in terms of Response Time and Makespan Time. Section 6 gives conclusion drawn from this paper.

Cloud Scheduling: The concept of cloud scheduling can be broadly classified into:

- **Dependent scheduling**

  If there are precedence orders existing in tasks, a task can only be scheduled after all its parent tasks are completed [5]. It is often known as workflow scheduling and it is harder to implement. Dependency between the tasks must be resolved dynamically.

- **Independent scheduling**

  Tasks are independent of each other and they can be scheduled in any sequence [6]. Order of execution does not matter and no task is dependent on any other task in the schedule for its input. It generally happens when the tasks are received from different users.

  Stochastic optimization algorithms are of two types: Heuristic algorithms (HA) and meta-heuristic algorithms (MHA).

  Optimization problems can either be deterministic or stochastic. Lot of research is being done in the field of stochastic scheduling algorithms for cloud computing. Stochastic algorithms can be classifies as:

- **Heuristic algorithm**

  Heuristic means to find or discover by means of trial and error. The general problem with heuristic algorithms is that there is no guarantee that optimal solutions are reached though quality solutions are found in a reasonable amount of time. These algorithms are problem dependent.

- **Meta-heuristic algorithm**

  The second generation of the optimisation methods is meta-heuristic proposed to solve more complex problems and very often provides better solutions than heuristic algorithms. Every meta-heuristic method consists of a group of search agents that explore the feasible region based on both randomization and some specified rules.
[2] RELATED WORK

Metaheuristic techniques are usually slower than deterministic algorithms and the generated solutions may not be optimal, most of the research done is towards improving the convergence speed and quality of the solution. [1]. Significant advances have been made in the nature-inspired computing (NIC) with a focus on the physics- and biology based approaches and algorithms. [4]. Goal Oriented Task Scheduling (GOTS) schemes give service providers a fair chance to apply approach and schedule the tasks and resources that can generate maximum possible economic gains, while using least resource provisioning. Using low resource provision allows providers to use their resources at possible fullest and trading Makespan with marginal increase only. [2]. Static scheduling algorithm like FCFS and Round Robin algorithms are fair algorithms but are outperformed by metaheuristic algorithm like ACO. [3]. Autonomous agent based load balancing algorithm (A2LB) is a dynamic resource scheduling algorithm that provides scalability and reliability by offering better resource utilization, and minimum response time. Whenever a VM becomes overloaded, the service provider distributes the resources in such a manner that the available resources are utilized in a proper manner and load at all the virtual machines will remain balanced. Migration ant in A2LB uses ants which are capable of choosing shortest/best path to their destination, which in case of task scheduling is selecting the best schedule [8].


A2LB is an ant based load balancing algorithm designed by Aarti Singh et al. in 2015. Autonomous agent based load balancing algorithm (A2LB) tries to address the issues like optimizing resource utilization, improving throughput, minimizing response time, dynamic resource scheduling with scalability and reliability. A2LB works by distributing the resources in such a manner that the available resources are utilized in a proper manner and load at all the virtual machines remain balanced. A2LB mechanism comprises of three agents: Load agent, Channel agent and Migration Agent. Load and channel agents are static agents whereas migration agent is an ant. The reason behind deploying ants is their ability to choose shortest/best path to their destination. Ant agents are motivated from biological ants which seek a path from their colonies to the food source. [8]. Following are the responsibilities of three agents:

Load Agent: The major responsibility of a load agent is to calculate the load on every available virtual machine after allocation of a new job in the data centre. It maintains all such information in table termed as VM_Load_Fitness table. [8].

Channel Agent: The channel agent initiates migration agents on receiving the request from load agent. The idea is to search for virtual machines with similar configuration in other data centres. It maintains the information received from migration agent in table termed as Response Table. [8].
Migration Agent (MA): Channel agent is responsible for initiating the Migration agent. Migration agent communicates with load agents of other data centers to find a compatible VM whose fitness value is greater than 25. [8].

In case any such VM is found, the channel agent migrates the task to that VM.

[4] ACO

ACO was introduced by Dorigo in 1992 as part of his Ph.D. work. ACO is a metaheuristic approach. It is based on the ability of ants to find the shortest path to food. Initially all ants follow random paths to their destination. Each ant leaves a chemical substance called pheromone. The intensity of pheromone evaporates with time. Initially the intensity of pheromone on each path is very less. As ants choose random paths to their destination, the intensity of pheromones starts to grow. With passage of time, the intensity of pheromones on longer paths evaporates much faster than the intensity of pheromones on shorter paths. Hence, smaller paths draw more pheromone and thus, pheromone intensity helps ants to recognize smaller paths to the food source. [10].

Following is the pseudo code for implementation of ACO. [10]

Algorithm 1: ACO algorithm

Input: List of Cloudlet (Tasks) and List of VMs
Output: The best solution for tasks allocation on VMs
Steps:
1. Initialize:
   - Set Current_iteration_t=1.
   - Set Current_optimal_solution=null.
   - Set Initial value τ_ij(t)=c for each path between tasks and VMs
2. Place m ants on the starting VMs randomly.
3. For k:=1 to m do
   - Place the starting VM of the k-th ant in tabuk.
   - Do ants_trip while all ants don't end their trips
   - Every ant chooses the VM for the next task according to Equation 1.
   - Insert the selected VM to tabuk.
   End Do
4. For k:=1 to m do
   - Compute the length L_k of the tour described by the k-th ant according to Equation 4.
   - Update the current_optimal_solution with the best founded solution.
5. For every edge (i, j), apply the local pheromone according to Equation 5.
6. Apply global pheromone update according to Equation 7.
7. Increment Current_iteration_t by one.
8. If (Current_iteration_t < tmax)
   - Empty all tabu lists.
   - Goto step 2
Else
   - Print current_optimal_solution.
End If

Dinesh Gupta, Harmaninder Jit Singh Sidhu and Gagan Kumar
Algorithm 2: 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

[5] RESULTS AND DISCUSSION

Parameters Setting of Cloudsim
The experiments are implemented with 3 Data Centers with 2 hosts each. 12VMs are created with 4 VMs per DC. Experiment was done with 200-500 tasks under the simulation platform. The length of the task is from 20000 Million Instructions (MI) to 400000 MI. The parameters setting of cloud simulator are shown in Table 1.

Table 1: Parameters Setting of Cloudsim

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task(Cloudlet)</td>
<td>Length of Task</td>
<td>20000-400000</td>
</tr>
<tr>
<td></td>
<td>Total Number of Task</td>
<td>200-500</td>
</tr>
<tr>
<td>Virtual Machine</td>
<td>Total Number of VMs</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>MIPS</td>
<td>256-512</td>
</tr>
<tr>
<td></td>
<td>VM Memory (RAM)</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>Bandwidth</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>Cloudlet Scheduler</td>
<td>Time_shared and Space_shared</td>
</tr>
<tr>
<td></td>
<td>Number of PEs Requirement</td>
<td>1-2</td>
</tr>
<tr>
<td>Data Centre</td>
<td>Number of Datacenter</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Number of Host</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>VM Scheduler</td>
<td>Time_shared and Space_shared</td>
</tr>
</tbody>
</table>
The Experiment was conducted by keeping the number of VM’s fixed. The arrival time for each task is considered to be 0 for the computation of response time. Hence the Execution Start Time of each cloudlet is its response time. Average response time is computed in the end by dividing the overall response time with the total number of tasks. Makespan is computed as the Finish Time of the last cloudlet. Following is the outcome of the experiment.

As you can see in Fig 1, the Response Time for both the algorithms is almost same. As the number of cloudlets increase, there is some variation in the results. It was also observed that the performance of A2LB varied quite a lot as the performance greatly depends on the initial random allocation of resources to the tasks.

As you can see in Fig 2, the performance of ACO is quite steady, whereas the performance of A2LB is unpredictable. Performance of both the algorithms is almost same even as you increase the cloudlets to 500.
[6] CONCLUSION

Performance of both the ant based algorithms in term of response time and Makespan time is quite similar. But the performance of A2LB greatly depends on how the initial allocation of resources to tasks is done. Although A2LB seems to perform slightly better, but it is more time consuming algorithm and it involves migration of tasks. The implementation of ACO in this paper does not perform any migration, whereas A2LB migrates the tasks from the overloaded VM to other VM. Also the A2LB is a resource aware scheduling algorithm, whereas ACO only considers the ETC for allocation the resources to tasks. Performance of ACO can be further improved by adding resource utilization parameters along with the ETC in its probability function. Further migration policy can also be added to ACO to migrate the tasks in queue and update the pheromone values.

REFERENCES


