ACO Based Task Scheduling Algorithm for Hybrid Cloud

M.Anuradha*1 and S.Selvakumar*2

*PG Scholar, Computer Science, GKM College of Engineering and Technology, Chennai, India.

* Ph.D, Computer Science, GKM College of Engineering and Technology, Chennai, India.

Abstract—Cloud computing is a subscription-based service where a network storage space and computation resources can be obtained. IaaS (Infrastructure as a Service) has become an important paradigm in cloud computing, it achieves economy of scale by multiplexing and therefore faces the challenge of scheduling task. The proposed scheduling approach, based on ACO (Ant Colony Optimization) schedules task from private cloud to external cloud when there is a peak demand. The system is modeled as a framework where a provider outsources its tasks to external cloud (Ecs) to meet the unpredictable situation in cloud environment. The approaches so far used rely on inter-cloud agreement that is required for the cloud federation. The proposed scheduler allocates the task to resource and adopts dynamic change in the cloud environment without any standardization also it provides optimized scheduling by which the factors of a good scheduler such as high throughput and reduced response time can be achieved and solves the issue of portability while migrating to different cloud. It was observed that the scheduling approach based on ACO yields better results and improves cloud provider’s profit by 0.26-11.58% compared with the existing methods.

Index Terms—Ant Colony Optimization (ACO), Hybrid cloud, Infrastructure as a service (IaaS), task scheduling.

I. INTRODUCTION

Cloud Computing provides scalable resources dynamically as a service over internet in order to assure lots of monetary benefits to be scattered among its adopters. According to Gartner cloud computing is defined as “a style of computing in which scalable and elastic IT-enabled capabilities are delivered as a service to external customers using Internet technologies.” According to Forrester cloud computing is defined as “a standardized IT capability (services, software, or infrastructure) delivered via internet technologies in a pay-per-use, self-service way.” Cloud-based services integrate globally distributed resources into seamless computing platforms.

Infrastructure as a Service (IaaS) becomes very popular as the foundation for higher level services such as Platform as a Service (PaaS) and Software as a Service (SaaS) [1]. IaaS providers such as Amazon EC2 and IBM Smart Cloud Enterprise [2], allow users to rent resources in the form of Virtual Machines (VMs). They can offer different VM types that are characterized by machine configuration, QoS[10] and pricing model.

A straightforward solution for a cloud provider is to over purchase the cloud resources such as memory, bandwidth in advance, which is not cost-efficient [4]. An other solution is Cloud Federation [3] that allows providers to trade their resources through federation agreement.

In this paradigm, providers can overcome their resource limitation by out-sourcing requests to other members in the federation. However, this federation is not easy to achieve at present, due to the lack of inter-operation standard and players’ motivation to federate [5].

To make an IaaS cloud itself elastic, a cloud resources allocation framework is proposed to allow it to utilize external clouds.

In this framework, an IaaS cloud has its own private cloud, and is able to outsource its tasks to other cloud providers called external clouds (ECs) when its local resources are not sufficient. An integer programming formulation is established, with the objective of maximizing the profit of the private cloud while at the same time guaranteeing QoS[10] to the user.

II. CHALLENGES AND REQUIREMENTS

There exists no generic model to represent various scenarios of task scheduling[16], especially when user’s requirements are vague and hard to encode through modeling languages. In particular, mapping QoS[10] requirements of applications to fine-grained resource level attributes is difficult. Modeling and quantifying non-functional requirements such as availability, is challenging.

Model parameterization i.e., finding suitable values for parameters in a proposed model, is a tedious task when the problem size is large. For example, for a multi-cloud scenario that includes n cloud providers and m VMs, m*n2 parameter assignments are needed in principle to express the VM migration overheads ignoring possible changes of VM sizes. Therefore, mechanisms that can help automatically capture those values are required.

The initial VM placement problem is typically formulated as a variant of the class constrained multiple-knapsack problem that is known to be NP-hard [16]. Thus to solve large-scale problem instances, tradeoffs between quality of solutions and execution time must be taken into account. This is a very important issue given the size of real life data centers.

Conflicting objectives. On energy-efficient scheduling, existing work focuses on certain aspects of QoS[10],
however they commonly overlook the energy-efficiency aspect that may conflict with other QoS requirements.

Continuous optimization[16]. Given the dynamic nature of clouds, resource allocation need to be renewed regularly for performance reasons, failure, etc., It is challenging to efficiently decide when and how to reconfigure the cloud in order to dynamically adapt to the changes.

III. SYSTEM ARCHITECTURE

Consumer layer is the point where the customers interact with the cloud service provider. The cloud user considered in the proposed system is a thick client. In addition to that there can be a thin client and mobile client.

In general consumer layer comprises various cloud user. Presentation layer allows to implement and design user interface through which the cloud user’s interact with service provider it also allows the management of user interaction which is required in systems where more than one user interact.

The Scheduler component allows to allocate suitable resources to user task by collecting details about the resource availability. It performs effective scheduling with the available cloud resources and in case of peak demand the tasks are being outsourced to the external cloud where the data center is identified by means of pricing models.

Resource pool in cloud environment allows the cloud service provider to serve multiple cloud consumers. The resources of a cloud environment includes storage, bandwidth and cpu. By providing processing, storage and network components as a resource pool the cloud computing environment allows the consumer to feel free to manage and control those resources.

Resource monitor provides information to the scheduler about the availability of resources. The scheduler interacts and collects data about resources for the effective utilization of resource so that no resource will be set as idle in the service provider.

The Cloud interface enables services to move between different providers and allows clients to easily switch between providers based on business objectives.

IV. SOLUTION FRAMEWORK

Consider[18] CP={CP₁,CP₂,…,CPₙ} be the set of cloud providers. Assume CP₁ is the private cloud and CP₂,…,CPₙ are external clouds. VM={VM₁,VM₂,…,VMᵢ} be the set of VM types and A={a₁,a₂,…,aₙ} be the set of applications required to be scheduled for each VM and each application has a runtime rᵢ and a task set {tᵢ₁,tᵢ₂,…,tᵢₗᵢ}.

An Integer programming model is formulated to solve this problem. The objective of integer programming formulation is to maximize the profit of private cloud on the premise of guaranteeing QoS. To formulate this problem, problem parameters and decision variables are defined in Table I and Table II, respectively.

The problem can be formulated as the following IP model. Maximum Profit achieved by the cloud service provider is

\[ \text{Maximize} \quad \sum_{f=1}^{F} \sum_{p=1}^{P} \sum_{v=1}^{V} \sum_{l=1}^{L} \sum_{k=1}^{K} y_{jlk} p_{v} v_{r} r_{f} \]

Subject to

\[ \sum_{l=1}^{L} y_{jlk} = 1, \quad v_{rf} \in \{1,2,\ldots,w\}, \quad f \in \{1,2,\ldots,T\} \]


<table>
<thead>
<tr>
<th>N</th>
<th>Number of cloud providers</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Number of VM types</td>
</tr>
<tr>
<td>W</td>
<td>Number of applications</td>
</tr>
<tr>
<td>P_v</td>
<td>Price of the vth VM</td>
</tr>
<tr>
<td>C_{kv}</td>
<td>Cost of the vth VM type in CP_k</td>
</tr>
<tr>
<td>R_{j}</td>
<td>Runtime of each task in the jth application</td>
</tr>
<tr>
<td>T_{l}</td>
<td>Number of tasks in the jth application</td>
</tr>
<tr>
<td>B_{kv}</td>
<td>If b_{kv}=1, the jk application use VM type VM_v, otherwise, it does not use this type.</td>
</tr>
</tbody>
</table>

Table I. Problem parameters.
Ant colony optimization is a probabilistic technique useful in problems which deal with finding better paths through graphs. Artificial or simulated agents locate optimal solutions by moving through a parametric space representing all possible solutions. Natural agents lay down pheromones directing each other to food source while exploring the environment. Similarly, ants record their positions and the quality of their solutions while migrating for resources. So that in later iterations more ants locate better solution by knowing the history of their predecessors. Here, simulated agents are considered as user’s task and the food source is cloud resources such CPU, Memory, etc.

In the proposed system, when there is a peak demand or massive unpredictable requests in the cloud environment, the cloud provider can outsource its requests which it cannot handle to one or more public clouds. In order to identify the suitable and optimal resources for the submitted user task, Ant colony based scheduling is proposed. The tasks are blind and migrated towards public cloud because of the insufficiency of provider. ACO provides a suitable and best solution in this problem instance like the original ants searching for food sources based on the pheromone trail. ACO broker is implemented in CloudSim[3], a simulator for cloud environment.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

Experiments were conducted using CloudSim[3], the cloud simulator. Initially user traffic were generated for the allocation of requests in private cloud. Round robin based scheduling approach were used for the scheduling in the private cloud.

In order to show insufficiency of resources in private cloud, more number of unpredictable requests were generated. ACO based scheduling is performed among the public cloud for the suitable and optimal allocation of user requests. As a result of simulation we observed that the proposed system improves the profit of cloud provider by satisfying the user who could not understand where their actual tasks are executed. And at the same time it provides QoS to the user where no user requests are rejected irrespective of the situation.

At normal workload the number of tasks varies gradually with time. The Fig 3 depicts a typical graph with increase in requirement of cloud resources. At usual workload the number of tasks to be handled by a cloud service provider gradually increase based on the requirements of the user. The user traffic will be predictable and predetermined. The virtual machines are identified based on the requirements of the user.

- Figure 3: Graph depicting normal workload
- Figure 4: Graph representing peak load
The proposed system addresses the issue of scheduling task in peak load. Where as the arrival of user requests are massive and unpredictable. In Fig 4, in a typical cloud environment the number of task are more at time 4 and hence the requirement for cloud resources will be maximum.

The provider has to satisfy the user request by providing sufficient resources for the execution of task in cloud environment. It is necessary that the request of the user should not be rejected, if it occurs the user may switch over to other provider hence causes performance degradation.

Some of the scenarios involved in the execution are tabulated below.

<table>
<thead>
<tr>
<th>Name</th>
<th>CPU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>1</td>
<td>1.8</td>
</tr>
<tr>
<td>Large</td>
<td>5</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Table III. VM Instance types

<table>
<thead>
<tr>
<th>Application</th>
<th>VM Instance types</th>
<th>Runtime</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Resources</td>
<td>Number</td>
</tr>
<tr>
<td>Number of tasks</td>
<td>10 batches</td>
<td>CPU</td>
<td>512</td>
</tr>
<tr>
<td>VM instance types</td>
<td>2</td>
<td>Memory</td>
<td>1024GB</td>
</tr>
<tr>
<td>Runtime</td>
<td>1 hour</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Throughput</td>
<td>5 batches</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table IV. Problem Instance

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average runtime</th>
<th>Average throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPO</td>
<td>4.4</td>
<td>500 bps</td>
</tr>
<tr>
<td>SLPSO</td>
<td>4.6</td>
<td>550 bps</td>
</tr>
<tr>
<td>Round Robin</td>
<td>4.9</td>
<td>570 bps</td>
</tr>
<tr>
<td>ACO</td>
<td>5</td>
<td>600 bps</td>
</tr>
</tbody>
</table>

Table V. Comparison with other algorithms

Experimental results of existing system [18] outperforms the standard PSO[6] for a given problem instance is given by actual best solution hence the provision of resource is not profitable for the provider. The user who requires the resource with RAM capacity of 100GB may provided with the resource with 120GB or 90GB which is not actual but nearer.

ACO outperforms the existing system by providing a better utilization of resources based on the value in the pheromone table and it randomly chooses the task in the intention that every task is important.

VII. RELATED WORK

In cloud computing[1], scheduling of user task in cloud environment is not the same as in traditional scheduling methods. Hence, task scheduling in cloud computing has focus and different methods have been proposed by cloud researchers.

The issue of scheduling task in cloud environment includes proposal for effective utilization of cloud resources and providing better service to the cloud user[19]. Some of the work that have been done in the past are discussed below.

In[12], Ruber etal., proposed task outsourcing in terms of heavy load for increasing the utilization of datacenter. But it failed to specify which workloads to outsource and where to schedule.

In[1], Ciu etal., proposed a resource reservation strategy in cloud environment which includes previous log records for reservation. The system considers only the single cloud and also rejects the user requests when resources are not sufficient.

Scheduling approaches based on genetic algorithm[5] considers independent and dividable tasks. The approach minimizes the maximum completion time of all tasks but does not considers the case of resource limitation also it suggests to use the resources as few as possible.
ACKNOWLEDGMENT

I thank the almighty for giving me this thinking and innovativeness in the research field and my family especially my son for his patience that I spent most of his time with me, in this work. I am very grateful to my guide Dr.S.Selvakumar for providing constant support throughout this work.

REFERENCES


S.Selvakumar received Doctor of Philosophy in Computer Science and Engineering from the Anna University. He has over 20 years experience in institutions and Organizations. Currently he is a Professor and Head of CSE, GKM College of Engineering & Technology. He is a recognized Supervisor for Ph.D of Anna University, Chennai and guiding 8 Ph.D Scholars. He has long been interested in Software Engineering. He has carried out various ARCTE sponsored Short Term Programs and worked on various Projects. He is a Senior Member in the Computer Society of India, Member in IEEE, ACM, the Institution of Engineers and the ISTE. He has published over 20 papers in International Journals and Conferences and reviewed journals and conferences. He is Authorized Trainer PSP/TSP by Software Engineers Institute (SEI), Carnegie Mellon University. He carried out training programs for Integra Microsystems Ltd, Tata Consultancy Services.