Grouping-Based Job Scheduling in Cloud computing using Ant Colony Framework

Madhusree Kuanr*, Prithviraj Mohanty**, Suresh Chandra Moharana***
*(Department of Computer Science, Vivekananda Institute of Technology, Bhubaneswar, INDIA)
** (Department of Information Technology, ITER, S'O'A UNIVERSITY, Bhubaneswar, INDIA)
*** (School of Computer Engineering, KIIT UNIVERSITY. Bhubaneswar, INDIA)

ABSTRACT

Cloud Computing is an emerging technology that provides computing resources on demand as per user requirements. It provides scalable resources needed for the applications hosted on it. However, applications submitted to cloud consist of varying length jobs according to their computational needs and other requirements. Although, it is a big challenge to design an efficient scheduler and its implementation, there exists some grouping based job scheduling strategy intends to minimize job transfer time, processing time and on the other hand maximizing resource utilization than without grouping based scheduling. Further analysis and research on job scheduling in cloud computing is necessary and desirable to enhance the performance of grouping based scheduling algorithm. In this work, we propose a grouped based approach for resource allocation in cloud for achieving the performance improvement by extending the concept of grouping based job scheduling using Ant Colony Framework with an objective to minimize the processing time and maximize the resource utilization.

Keywords - **Ant Colony Framework, Cloud-computing, Job-grouping, Job-scheduling, Resource Allocation.**

I. INTRODUCTION

Cloud computing which provides cheap and pay-as-you-go computing resources is getting popular and acts as an alternative to traditional IT Infrastructure. The goal of resource management in cloud is that resources must be available and meet performance criteria towards requests. A cloud-based computing resource is thought to execute or reside somewhere on the "cloud", which may be an internal corporate network or the public internet. From the perspective of an application developer or information technology administrator, cloud computing enables the development and deployment of applications that exhibit scalability, performance and reliability, all without any regard for the nature or location of the underlying infrastructure [1].

Though there are various advantages in cloud computing, the major concern is the order in which the requests are satisfied. This evolves the scheduling of resources and jobs. The resources must be allocated efficiently to maximize the resource utilization and minimize the processing time [2].

The cloud resources are dynamic, heterogeneous and of different processing capabilities. An application with large number of varying-length independent jobs when submitted individually to the cloud resources incurs a communication overhead that may be more than the total amount of computation time of each job at the resource. In addition, this also leads to poor utilization of the resources. Therefore, jobs can be grouped at the scheduling level according to the processing capabilities of the available resources. In this paper, we present a group based mechanism for resource allocation in cloud computing environments for hosting the applications with given QoS requirements using different Ant Agents.

The rest of the paper is organized as follows. In the next section we have surveyed some works related to grouping based job scheduling and Ant Colony Framework for job scheduling. Section 3 describes the system architecture, our methodology and algorithms used for the resource allocation. Section 4 outlines the simulation results and section 5 represents the conclusion and the future work.

II. RELATED WORK

Resource allocation for clouds has been studied very extensively in the literature. The problem of deciding on an optimal assignment of requests to resources allocator is NP-hard [2]. Several heuristic algorithms have been proposed by researchers for optimal allocation of cloud resources. Dynamic Job Grouping-Based Scheduling in Grid Computing includes a dynamic job grouping-based scheduling algorithm that groups the jobs according to MIPS of the resource [3]. It selects resources in first come first serve order and multiplies the granularity size to increase the resource computation time. The main advantages of this type of scheduling is, it reduces the total processing time and maximizes the utilization of the resource. Dynamic

resource characteristic constraints like bandwidth and memory size are not taken into account while scheduling the jobs.

Grouping-based Fine-grained Job Scheduling in Grid Computing focuses on lightweight job scheduling [4]. In Grid computing there exists several applications with a large number of lightweight jobs. Sending some fine grained jobs to a resource that can support high processing capability is not economical compared with sending a coarse-grained job. It defines how fine grained jobs are grouped into coarse grained jobs and how they are allocated.

A Bandwidth-Aware Job Grouping-Based scheduling strategy schedules the jobs with MIPS and bandwidth of the resource [5]. The principle behind the bandwidth-aware scheduling is the scheduling priorities taking into consideration not only their computational capabilities but also the communication capabilities of the resources. This strategy reduces the processing time of the jobs and maximizes the utilization of the resource.

The proposed grouping based resource allocation strategy in Memoryaware Dynamic Job Scheduling Model in Grid Computing is used to maximize the resource utilization and minimize the processing time of the jobs [6]. The job scheduling is based on job grouping concept taking into account memory constraint together with other constraints such as processing power, bandwidth, expected execution and transfer time requirements of each job. These constraints are taken at job level rather than at group level.

A Time-Minimization Dynamic Grouping-Based Job Scheduling in Grid Computing presents and evaluates an extension to dynamic job grouping-based job grouping scheduling strategy that maximizes the utilization of Grid resource processing capabilities, and reduces processing time and communication time and execute the jobs on the Grid [7]. The model converts the fine-grained jobs into the coarse-grained job or grouped job according to jobs requirement and available resource capability.

The resource allocation strategy used in Power efficient resource allocation for clouds using ant colony framework focuses on the Quality of services(QoS) constrained resource allocation problem in which customers are willing to host their applications on the provider's cloud with a given Service Level Agreement(SLA) requirements for performance such as throughput and response time [8]. It proposes energy efficient and agent based mechanism that allocates the cloud resources to the applications without violating the given SLA using Ant colony framework. This mechanism is very flexible and can be extended with improvements, as the solution modules are modeled as independent intelligent agents.

Cloud Computing Initiative using Modified Ant Colony Framework proposes a heuristic algorithm based on modified ant colony optimization has been proposed to initiate the service load distribution under cloud computing architecture [9].In this paper, the pheromone update mechanism of ACO and coefficient is modified. This modification supports to minimize the makespan of the cloud computing based services and probability of servicing the request also has been converged using the modified scheduling.

The efficient scheduling of independent jobs in a heterogeneous computing environment is an important problem in domains such as grid computing. The proposed Ant colony algorithm has a modified pheromone updating rule which solves the grid scheduling problem effectively than that of the existing Ant colony algorithm [10].

III. PROPOSED SYSTEM ARCHITECTURE

The main aim of our resource allocation is to allocate resources for applications oriented jobs. The resource allocation strategy is used for satisfying the service requests of users, we use the following architecture.

Users/Brokers: Users or brokers acting on their behalf submit service requests from anywhere in the world to the cloud via cloud controller for processing. They provide organizations with a single point of entry into multiple clouds.

Cloud Controller: The Cloud Controller is responsible for exposing and managing the underlying virtualized resources (machines (servers), network, and storage) via user-facing APIs. It acts as the interface between the users and the cloud service providers.

Virtual Machines(VMs): A virtual machine (VM) is a software implementation of a computing environment in which an operating system or program can be installed and run. The virtual machine typically emulates a physical computing environment, but requests for CPU, memory, hard disk, network and other hardware resources are managed by a virtualization layer which translates these requests to the underlying physical hardware.The VMs can host multiple operating system environments which are completely isolated from one another on the same physical machine.

Physical Machines: These are the physical computing servers that will provide hardware infrastructure for creating virtual machines.

Available Resource Table: The information of the resources consisting of Processing Power (MIPS), Available time, and Memory Capacity are gathered before admitting the resources into the cloud. We sort the nodes according to decreasing order of their

processing power. Then we store the nodes consisting of Node Id, Processing Power (MIPS), Memory and Available time in the Available Resource Table. We have assumed that all the nodes are having the same network connectivity. The allocation pointer of the Available Resource Table will be adjusted by different ant agents.

We have followed the terminology related to different types of ant agents used in [10].The functionality of these ant agents is explained in the following subsections.

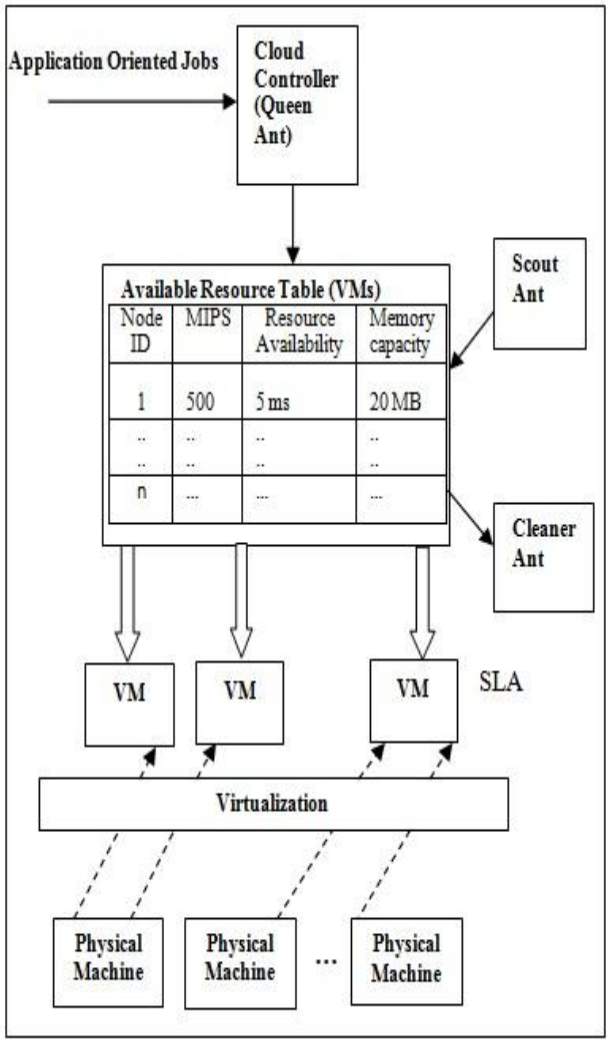

Fig.1 Proposed System Architecture

Cloud Controller and Queen Ant

The jobs from the user consisting of job size, application code are given to the Cloud Controller. We have treated these service requests as jobs having specific computational power (MI). The Cloud controller maintains a queue (Q) for storing these jobs and also sorts the jobs according to the decreasing order of their computational power (MI). Then it groups the jobs depending upon the MI of the jobs and processing power (MIPS) of the resource to which the allocation pointer of the resource table is pointing to. Grouping of jobs in this paper is based on processing capability (in MIPS) of available resources. Jobs are grouped according to the capability of the selected resource. Therefore, the following conditions must be satisfied:

$$\text{Groupedjob}_{\text{MI}}() \leq \text{Resource}_{\text{MIPS}} * \text{Avail. Time} \quad (1)$$

$$\text{Groupedjob}_{\text{MI}} = \sum_{i=1}^{k} \text{JOB}_i \text{ where } k \geq 1 \quad (2)$$

Where, MI (Million Instruction) is job's required computational power, MIPS (Million Instruction per Second) is processing capability of the resource and Available time is resource's availability time in the cloud.

The Grouping Strategy is as follows:

The resources and jobs are sorted in descending order of their processing power (MIPS) and job length (MI) respectively. The resources are taken from the Available Resource Table and the allocation pointer in the resource table points to the resource (node) on which next service request is deployed. Once selecting a resource in this manner, jobs are added into job group according to the processing capability of this resource by alternatively taking jobs from front end i.e. job with higher length and then rear end of the job list i.e. job with smaller length. It should be noted that the front end and rear end pointers are updated to point to the next job in the list. While grouping the jobs taken in order from the job list, the above grouping strategy falls into one of the two cases given below.

Case1. At some stage in grouping, if given condition in equation 1 fails, while adding a job into the group from front end of the job list, then it is removed from the group and if possible jobs are taken from the rear end of the list till it satisfies the equation1.

Case2. If condition in equation 1 fails while adding a job from rear end of the job list during the grouping operation, then it stops grouping for that resource and sends the job group to the dispatcher. Then, it takes next highest resource to perform another job grouping. The above job grouping process is repeated for each resource taken in given order as long as either resources or jobs exist.

A node (VM) is to be considered as a valid node with minimum processing power, minimum memory and minimum bandwidth decided by the broker.

The Queen ant is responsible for sorting and grouping of the jobs as shown in Algorithm1.

Algorithm 1

1. Sort the jobs in the JobList[] according to the decreasing order of their job length (MI).
2. If ResourceList[] pointer currently points to a valid node

BEGIN

While($\text{JobGroup}_{\text{MI}} \leq \text{Node}_{\text{MIPS}} * \text{Availabletime}$)

2.1 Group the jobs by taking the big job from the left and small job from the right side of the JobList[] alternatively.

END

Worker Ant

The worker ant is responsible for allocation of the job group to the node. So the worker ant is always looking for a pending request (job group) in the queue to be processed. If it finds any such request, it dequeues the request and calls the Algorithm 2.

Algorithm 2

1. If ResourceList[] pointer currently points to a valid node

BEGIN

1.1 Call Algorithm-1[For grouping of jobs]

1.2 If($JobGroup_{MI} \leq Node_{MI}$) and ($JobGroup_{Memory} \leq Node_{Memory}$)

BEGIN

-Deploy the JobGroup on a node

-Creates SLA monitor agent to monitor it.

END

1.3 Else if($JobGroup_{MI} \leq Node_{MI}$) and ($JobGroup_{Memory} \geq Node_{Memory}$)

-Divide the JobGroup and allocate to different nodes (VM).

1.4 Else

- Reject the request.

END

2. Else

- Reject the request.

The VMs created like Amazon Standard Instance with specific CPU processing power and memory [11]. We assumed that the load is balanced in every node in the cloud.

SLA Monitor Agent

It continuously monitors the hosted applications (job group) and calculates the average response time and throughput of the job group. Then it passes this information to the SLA Resource Allocator.

Scout Ant

The job of the scout ant is to discover new cloud nodes having processing power (MIPS) and memory as shown in Algorithm 3. It does through the node registration [8].

Algorithm 3

1. Scout Ant visits each node
2. For each node if a new node sends a request to this node to join

-If it is a valid node

BEGIN

-Register the node

-Update its information in the resource table

END

-Else

BEGIN

-Reject the request

END

In the node registration mechanism, the nodes those who want to join in the cloud sends request to one of the nodes in the cloud. When the Scout Ant finds any such request while visiting the node, it registers that node with a unique id. Then it places this node in its appropriate position in the resource table with the required information (MIPS, memory, Available time).We have assumed that the node that is completing the registration for newly joining node will not get failed during the registration process. If it fails before registration, new node will contact another node in cloud [8].

Cleaner Ant

It maintains the available resource table. It removes the unavailable resources from the list. When the cloud ant didn't get any response within a specific time while visiting that node, it assumes that this node is failed and it takes necessary actions for recovery. It will remove this node information from the available resource table. It also removes unused nodes which are underutilized. It removes the nodes whose service level agreement gets expired. It also sends the alert to the customer for the renewal of the service requests if they needed [9]. The cleaner ant visits each node in the cloud and performs the action as coded in Algorithm 4.

Algorithm 4

1. If the current resource consumption is NIL

BEGIN

-Remove this VM from this host.

END

2. If Processing time and Throughput are 30 percent more than that of SLA values

BEGIN

-Remove this VM from this host

END

3. If a VM lease time remaining is less than a week

BEGIN

-Notify User(Service Time is about to complete)

END

4. If a VM service time is over

BEGIN

-Remove VM from the node

END

SLA Resource Allocator

The SLA Resource Allocator acts as the interface between the Cloud service provider and external users/brokers. It contains Service Request Examiner and Admission Control mechanism which first interprets each submitted request for QoS requirements. Then they determine whether to accept or reject the requests. It makes service level agreements for accepted request. The working of the SLA Resource Allocator can be summarized by the Algorithm 5.

Algorithm 5

1. When a service request is first submitted, the service request examiner and admission control

mechanism interprets the submitted request for QoS requirements.

-If QoS requirements are satisfied
BEGIN
-It determines resource entitlements for allocated VMs by finding SLA values for ProcessingTime and Throughput.
END
-Else
BEGIN
-Reject the request.
END

Time Complexity Analysis: Considering the number of jobs are *n* and the number of resources are *m*, the time complexity for the Algorithm -1 will be O(nlg n) as we will follow the sorting of the jobs first.

IV. SIMULATION RESULTS

We have used the CloudSim [12] toolkit to simulate the proposed approach for group based resource allocation in cloud computing. The machine we have used for simulation has Intel Core 2 Duo 2.10 GHz processor and 3 GB of RAM. We have used Ubuntu 9.10 operating system and Java 1.6 for running the simulation code. We have used the ant tool to compile and create jar files for incorporating our modifications in the CloudSim framework. We have simulated the datacenter by creating a datacenter in CloudSim. Then, we have added a set of hosts to this datacenter. After that, we have created a set of virtual machines on top of it.
The simulation is done by assuming the average length of jobs as 100 and 200 MI. The length of jobs may be deviated to 30% . The average memory size of each job is assumed as 5 MB and the Available time is assumed as 10 second. The resource table used for simulation work is as follows.

TABLE 1. Available Resource Table

Node Id	Processing Power(MIPS)	Memory(MB)
1	300	60
2	250	48
3	200	40
4	150	32
5	100	25

We have analyzed the performance of the proposed approach by the two parameters, processing time and resource utilization. To evaluate the processing time of an application, an analytical performance model is defined in terms of overhead time and communication time of the jobs.
Let T_{PST} be the processing time, T_{OHT} be the overhead time and T_{COMM} be the communication time of a job.
Therefore:

$$T_{PST} = T_{OHT} + T_{COMM} \quad (3)$$

The overhead time T_{OHT} of a job is the waiting time for the resource. Communication time $T_{COMM}$is equal to the time taken to submit the job to the resource and getting results after processing the jobs.

Considering the average length of the jobs as 100 MI, fig. 2 represents the processing time of jobs for both grouping and non grouping strategies. Similarly fig. 3 depicts the processing time of jobs in both the strategies i.e. grouping and non grouping using average job length of 200 MI and available time of 10secs.We have concluded from the graph that the processing time for the grouped job is less than the processing time for the non-grouped jobs.

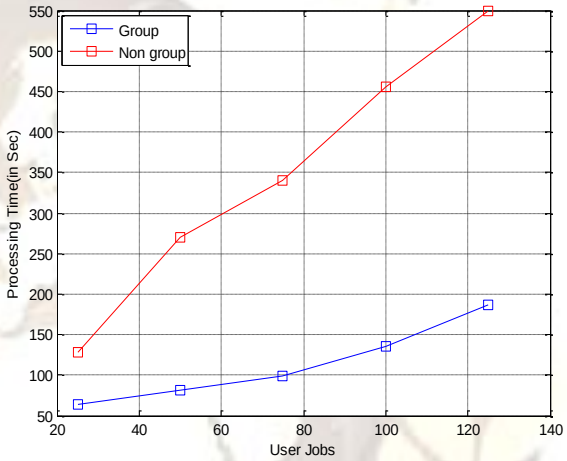

Fig. 2 Processing Time (Avg. MI=100)

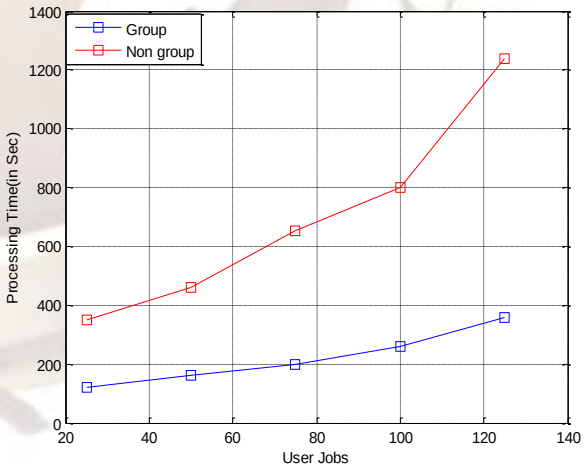

Fig. 3 Processing Time (Avg. MI=200)

The Available Resource Table maintains the list of available virtual machines along with their capacity and available memory. The job groups are assigned to the best possible virtual machine available in cloud. So, the processing power (MIPS) as well as

the memory of the virtual machine is utilized properly.
Therefore:

$$ResourceUtilization = JobSize / RessourceCapacity \quad (4)$$

$$RessourceCapacity = MIPS * Avail\ Time \quad (5)$$

We have also found out from the graph (fig. 4 and fig. 5) that the strategy of allocating grouped jobs to the virtual machines being leads to efficient utilization of resources (virtual machines) in cloud compared to non-grouped jobs.

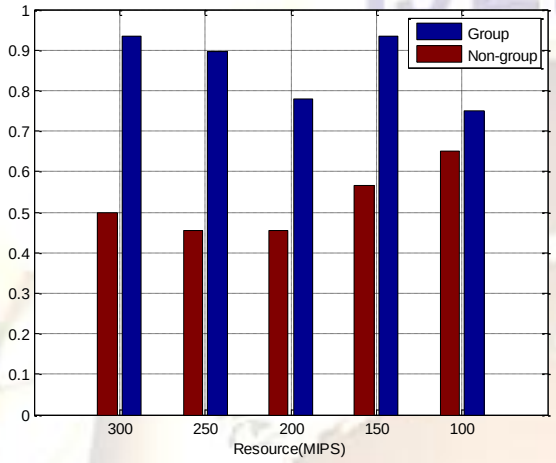

Fig. 4 Resource Utilization (Avg. MI=100)

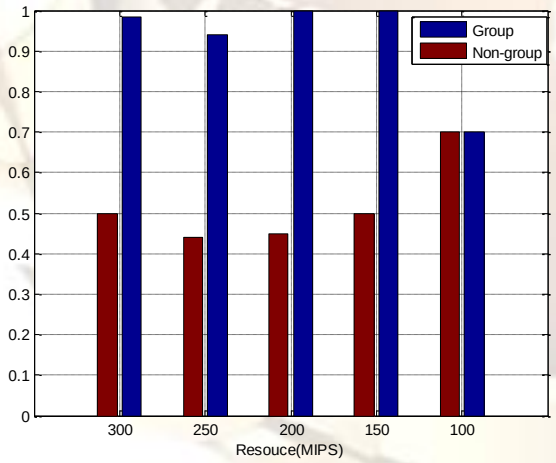

Fig. 5 Resource Utilization (Avg. MI=200)

V. CONCLUSION AND FUTURE WORK

The proposed Grouping-based job scheduling strategy used to allocate jobs to the available resources in a cloud environment reduces the processing time and maximizes the resource utilization. The Ant Colony Framework is used to deploy our approach in cloud. The individual loads for the resource may be considered while grouping the jobs. In this approach, the power consumed by the entire system is also a matter of concern. Failure of any nodes during the job execution is not considered at all. In future, we will modify our proposed approach taking the power consumption of the entire system, individual loads and fault tolerant of the resources.

References

[1] Gruman, Galen . What cloud computing really means. http://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-%means-031.

[2] B. Soumya, M. Indrajit, and P. Mahanti, "Cloud computing initiative using modified ant colony framework," in In the World Academy of Science, Engineering and Technology 56, 2009.

[3] N. Muthuvelu and Junyan Liu and N.L Soe and S. Venugopal and A. Sulistio and R. Buyya. A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained tasks on Global grods. In Proc of Australian workshop on grid computing, 4:41–48, 2005.

[4] Quan Liu and Yeqing Liao. Grouping-Based Fine-Grained Job Scheduling in Grid Computing. IEEE First International Workshop on Educational technology and Computer Science, 1:556–559, 2009.

[5] T.F Ang and W.K Ng and T.C Ling. A Bandwidth-Aware Job Grouping-Based Scheduling on Grid Environment. Information Technology Journal, 8(3):372-377

[6] M.K Mishra and R.Sharma and V.K Soni and B.R Parida and R.K Das. A Memoryaware Dynamic Job Scheduling Model in Grid Computing. In *ICCDA,2010 International Conference*, volume 1, pages 545–549, 2010.

[7] M. K. Mishra, P. Mohanty, and G. B. Mund. A time-minimization dynamic jobgrouping-based scheduling in grid computing. International Journal of Computer Applications, 40(16):16–25, 2012.

[8] L. Chimakurthi and M. K. S. D. Power efficient resource allocation for clouds using ant colony framework. 2011

[9] Soumya Banerjee, Indrajit Mukherjee, and P.K. Mahanti. Cloud Computing Initiative using Modified Ant Colony Framework. *World Academy of Science, Engineering and Technology* , 2009.

[10] Y. Li, "A bio-inspired adaptive job scheduling mechanism on a computational grid," vol. 6, Mar. 2006.

[11] "http://aws.amazon.com/ec2/," accessed on Nov 23, 2010.

[12] R.Buyya, R.Ranjan, and R.N.Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities.*IEEE Computer Society*, 2009