A New Task Scheduling LCGA Algorithm in Cloud Computing Environment

1A. Priyalatha, 2A. Ramathilagam
1PG Student, 2Associate Professor
Department of CSE,
P.S.R.Engineering College, Sivakasi

Abstract: Although some cloud providers look at the hybrid cloud as blasphemy, there are strong reasons for them to adopt it. Hybrid clouds offer the cost and scale benefits of public clouds while also offering the security and control of private clouds. Task scheduling, one of the most famous combinational optimization problems, plays a key role in the hybrid cloud environment. We propose a graph-based task scheduling algorithm. In order to achieve minimum cost, our algorithm takes into account not only the private resources but also the public resources. This paper also presents an extensive evaluation study and demonstrates that our proposed algorithms minimize the user’s cost in a hybrid cloud environment.

Keywords: Component: Cloud computing; Hybrid cloud; Task scheduling; Resource Allocation

I.INTRODUCTION
Recently, there has been a dramatic increase in the popularity of cloud computing [1] system (e.g., Amazon’s EC2, Google’s Google App Engine and Rackspace Cloud). According to [2], cloud computing has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware and purchased. By providing virtualized computing resources as a service in a pay-as-you-go manner, cloud computing enables new business models and cost effective resource usage. Instead of having to maintain their own data center, companies can concentrate their core business and purchase resource when needed. Especially when combining a privately maintained virtual infrastructure with publicly accessible clouds in a hybrid cloud [3], the technology can open up new opportunities for businesses and help consolidating resource. Even though hybrid clouds offer a great value proposition and enable many opportunities, the number of challenges and issues is also very high. Task scheduling, one of the most famous combinational optimization problems, plays a key role in the hybrid cloud environment. In general, Task scheduling is the process of mapping tasks to available resources on the basis of tasks’ characteristics and requirements [4]. It is an important aspect in efficient working of cloud as various task parameters need to be taken into account for appropriate scheduling. The available resources should be utilized efficiently without affecting the service parameters of cloud. In this paper, we propose a new task scheduling algorithm in hybrid cloud environment. We take into account not only the private resources but also the public resources. The results of experiments show that our algorithm is promising. The rest of this paper is organized as follows: Section II briefly discusses related work. Next, Section III describes the Eclipse Toolkit and the scheduling policies in it. Section IV presents the proposed scheduling algorithm. In Section V the experimental details and results of experiments are presented with comparison with some existing algorithms. Finally, Section VI concludes the paper and proposes future improvements.

II.RELATEDWORK
Now, there are several proposed task scheduling approaches for Hybrid Cloud, with different motivations and incentives. Zhang et al. [5] presented an intelligent workload factoring for a hybrid cloud computing model. It splits the system workload into two parts, base load and trespassing load. Base load can handle by private cloud, and public cloud service can deal with the trespassing load. They model the general workload factoring process as a hypergraph partition problem. This model can reduce data cache/replication overhead, but not support real-time QoS constraint computing. Song et al. [6] presented the genetic algorithms in virtualized environments. However, the genetic algorithms generally require a long execution time. The long execution time increases the probability of SLA violation in the Cloud computing environments, where customers need to be served immediately. Fu Y. et al. [7] proposed an SLA-based dynamic scheduling algorithm (Squeeze) of distributed resources for streaming. Moreover, Yarmolenko V. et al. [8] evaluated various SLA based scheduling heuristics on parallel computing resources using resource (number of CPU nodes) utilization and income as evaluation metrics. Our work focuses on task scheduling method in hybrid clouds. Popovici et al. [9] mainly considered QoS parameters on the resource provider’s side such as price and offered load, but didnot focus on the user side. However, our proposed work focuses on user driven scenarios. Wu et al. [10] proposed resource allocation algorithms for SaaS providers who want to minimize infrastructure cost and SLA violations. They implemented three cost driven algorithms which considered various QoS parameters (such as arrival rate, service initiation time and penalty rate) from both the customers’ and the SaaS providers’ perspective. Aneka [11] is a cloud application platform capable of provisioning resources obtained from a variety of sources, including private and public clouds, clusters, grids, and desktop grids. Christian et al presented Aneka’s deadline-driven provisioning mechanism, which is responsible for supporting quality of service (QoS)-aware execution of scientific applications in hybrid clouds composed of resources obtained from a variety of sources.
III. ECLIPSE TOOLKIT

Cloud computing aims to power the next generation data centers and enables application service providers to lease data center capabilities for deploying applications depending on user QoS (Quality of Service) requirements. Because cloud applications have different composition, configuration, and deployment requirements, so quantifying the performance of resource allocation policies and application scheduling algorithms in cloud computing environments is a challenging to tackle. To simplify this process, Buyya et al. [12] proposed a software named Eclipse which is an extensible simulation toolkit that enabled modeling and simulation of cloud computing environments. A. Eclipse Architecture Figure 1 shows the multi-layered design of the Eclipse software framework and its architectural components [12]. Initial releases of Eclipse before 2.0 used SimJava as the discrete event simulation engine that supports several core functionalities, such as queuing and processing of events, creation of Cloud system entities (services, host, data center, broker, VMs), communication between components, and management of the simulation clock. However, after version 2.0, the SimJava layer has been removed in order to allow some advanced operations that are not supported by it. The Eclipse simulation layer provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for virtual machines (VMs), memory, storage, and bandwidth. Eclipse layer manages the creation and execution of core entities (VMs, hosts, Data centers, application) during the simulation period. This layer handles the provisioning of hosts to Vs on the basis of user requests, managing application execution, and dynamic monitoring. The topmost layer in the Eclipse stack is the User Code that exposes basic entities for hosts (number of users and their application types, and broker scheduling policies). A Cloud application developer can write an application configurations and scenarios at this layer to perform a cloud computing scenario simulations.

![Layered Eclipse architecture](image)

Figure 1. Layered Eclipse architecture

B. Cloud Reports

Cloud Reports [13] is a graphic tool that simulates distributed computing environments based on the Cloud Computing paradigm. It uses Eclipse as its simulation engine and provides an easy-to-use user interface, report generation features and creation of extensions in a plugin fashion. The application simulates an Infrastructure as a Service (IaaS) provider with an arbitrary number of. Each data center is entirely customizable. The user can easily set the amount of computational nodes (hosts) and their resource configuration, which includes processing capacity, amount of RAM, available bandwidth, power consumption and scheduling algorithms. The customers of the IaaS provider are also simulated and entirely customizable. The user can set the number of virtual machines each customer owns, a broker responsible for allocating these virtual machines and resource consumption algorithms. Each virtual machine has its own configuration that consists of its hypervisor, image size, scheduling algorithms for tasks (here known as cloudlets) and required processing capacity, RAM and bandwidth. Additionally, CloudReports generates HTML reports of each simulation and raw data files that can be easily imported by third-party applications such as Octave or MATLAB. Figure 2 shows the screenshot of Cloud Reports’s GUI:
C. Schedule Policies in Eclipse

Figure 2 shows the two levels scheduling in Eclipse [14]. Eclipse models scheduling of CPU resources at two levels: Host and VM. At Host level, the host shares fractions of each processor element (PE) to each VM running on it. Because resources are shared among VMs, this scheduler is called Vm Scheduler. The scheduler a host uses is a parameter of the Host constructor. In the VM level, each virtual machine divides the resources received from the host among Cloudlets running on it. Because in this level resources are shared among Cloudlets, this scheduler is called Cloudlet Scheduler. The scheduler a VM uses is a parameter of its constructor. The Vm Scheduler models the behavior of scheduling at virtual machine level like VMMs such as Xen and VMware. Similarly, Cloudlet Scheduler models the behavior of scheduling at operating system level. Eclipse us Cloudlet Scheduler models to divide available resources among the applications running in the system. In this paper we focus on task scheduling. So we extended Cloudlet Scheduler class to model task schedule behavior.

IV. A GRAPH-BASED TASK SCHEDULE ALGORITHM

Task scheduling process in cloud can be generalized into three stages namely: 1. Resource discovering and filtering: Data center Broker discovers the resources present in the network system and collects status information related to them. 2. Resource selection: Target resource is selected based on certain parameters of task and resource. This is deciding stage. This paper focuses on this stage, and proposes a new approach to allocate resources to task effectively. 3. Task submission: Task is submitted to resource selected. To allocate available resources among applications effectively, we use a graph-based algorithm (Figure 3). Firstly we need to build a bipartite graph. A bipartite graph is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V; that is, U and V are independent sets. In our paper, U denotes task collection, and V denotes public vm or private vm collection.
Definition: a bipartite graph $G = (U, V, E)$ is a simple graph defined as follows: - $U$ is the set of vertices which denotes a public VM or a private VM. - $V$ is the set of vertices which denotes a task. - $E$ is the set of edges which all go between the X and Y. The weight of the edge is the VM cost of a discrete task. Cost of Task = (Cloudlet Length MI / vm MIPS) * unit cost of vm Time of Task = (Cloudlet Length MI / vm MIPS) The question is to find a match $\subseteq EM$ such that $\epsilon = Me ewMw )()$ is the minimum. To solve the minimum bipartite match problem, we use the Hopcroft-Karp algorithm [15]. This algorithm was found by John Hopcroft and Richard Karp. As in previous methods for matching such as the Hungarian algorithm and the work of Edmonds [16], the Hopcroft–Karp algorithm repeatedly increases the size of a partial matching by finding augmenting paths. However, instead of finding just a single augmenting path per iteration, the algorithm finds a maximal set of shortest augmenting paths. So, the complexity of our algorithm is $EVO ||(\text{ or } VO 5.2 )|]$, where $V$ is the number of the vertices of the bipartite graph. V. STUDY OF PERFORMANCES OF THE ALGORITHMS PROPOSED In our experiment, we use Cloud Reports and Eclipse 3.0 based on Java Platform, Standard Edition Development Kit (JDK) 7U0, running on a machine with Core 2 Duo CPU, E7500 @ 2.93GHz, 2GB RAM. To simulate our algorithm, we need to set up the configuration of tasks and VMs. Task Configuration. We build up some tasks (Cloudlet) in Cloud Reports (see Figure 4).

Figure 4. Cloudlet in CloudReports VMs Configuration.

In experiment 1, we build up a hybrid cloud environment with two kinds of data centers environment in CloudReports. (Figure 5) One represents public cloud, and the other represents private cloud. In experiment 2, we build up a cloud with one kind of data center. These data centers have 9,600MIPS computation ability, 40GB memory, 11TB storage and 10 GB bandwidth, with x86 architecture, Linux operating system and Xen virtual machines.
The simulation experiments were run under the following system configuration: (1) a hybrid clouds is available, hence there are two kinds of data centers: public cloud and private cloud. To use the private cloud resources effectively, we let the cost of private resources is zero; and (2) there is only one kind of data center (without hybrid cloud). Table 1 shows the average turn-around time for each Cloudlet and the overall cost of the user application for both cases. A user application consists of one or more Cloudlets with sequential dependencies. The simulation results reveal that in hybrid clouds our graph-based algorithm reduces the cost by more than 30%, while take a little time later by 12%. Table 1 Performance Results

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>With Hybrid Cloud</th>
<th>Without Hybrid Cloud</th>
<th>% Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Turn Around Time (Secs)</td>
<td>4283.3</td>
<td>3120.1</td>
<td>30%</td>
</tr>
</tbody>
</table>

V. CONCLUSION

Cloud computing provides four deployment models: public cloud, private cloud, community cloud and hybrid cloud. For an economical and efficiency way, hybrid cloud becomes an important environment. This paper focuses on task scheduling in hybrid cloud environment with the explicit aim of resources cost minimization. To achieve this goal, we propose a graph-based task scheduling algorithm which considers resources from both the private clouds and public clouds. The simulation results show that on average, our algorithm optimizes cost savings better when compared to the other proposed algorithms. In building on the research undertaken in this paper in the future, we propose to take into account other kind of resources in the task scheduling, such as disk storage and network bandwidth. Moreover, we will consider some QoS parameters (such as arrival rate and service initiation) to improve customer satisfaction levels.

REFERENCES


