Abstract

Cloud Computing is used to avail internet based services on a lease basis in a very short time period. Purchase and installation of new computing resources takes time, so getting resources as a service from cloud in a minute or so is faster and cheaper nowadays. A contract called service level agreement is worked between cloud host and consumer. SLA includes quality of service (QoS) that must be provided by the cloud host to consumer. It also includes time frame, in that time period consumers’ task should get done. Due to multi-tenancy architecture a cloud host has many consumers at same time. Some consumers cannot get their job done in time limit. When rejecting consumers’ leases, deadline plays an important role. This paper introduces mEDF (Modified Earliest Deadline First) algorithm and leasing policy for deadline driven resource management, which takes care of minimum capacity expenses at cloud host side. Experiments and results show that this algorithm is suitable and can be used for deadline driven scheduling with minimizing response time and completion time.

Keywords: Deadline, Resource management, IaaS cloud, mEDF.

1. INTRODUCTION

Cloud computing is the practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer [1]. These remote servers show infinite capacity. Resources provided by cloud can be increased or decreased on demand. Consumers have to pay only for what they have used. Proper monitoring of SLA is done to provide QoS to consumers.

Clouds can be public, private or hybrid. In a public cloud many consumers from same or different organization can be served at a time. In a private cloud, consumers only from the same organization are served from the cloud. Hybrid model of cloud allows some services from its own resources and some from other cloud providers. Cloud Computing provides three different types of services namely (1) Software as a Service (SaaS), (2) Platform as a Service (PaaS), and (3) Infrastructure as a Service (IaaS).

In SaaS model of cloud, QoS-based workflow scheduling algorithm based on Partial Critical Paths (PCP) can be used [2]. PCP tries to minimize the cost of workflow execution while meeting a user-defined deadline. Recursively scheduling the partial critical paths ending at previously scheduled tasks can also be done.

User defined deadline can also be provided in IaaS clouds. It may be a situation that a consumer’s task is to run in an unattended mode with a predefined deadline that either provide resources in a given time frame or reject the request. Haizea, which can be used as a plug-in scheduler for opennebula provides deadline driven scheduling [3]. This deadline driven policy does not account for deadline sensitivity with minimizing computing capacity requirements to cloud host.

Our proposed policy and algorithm minimizes computing capacity requirements while keeping earliest deadline meeting on the priority. Algorithm to minimize computing capacity requirement in deadline driven scheduling for different consumers tasks are proposed in this paper. Experimental results are also shown to advocate the algorithm.

Rest of the paper is organized as: Section 2 contains related work. Section 3 describes need of introducing mEDF algorithm for completing consumers’ task. In section 4, proposed algorithm, mEDF algorithm to minimize computing capacity while meeting deadlines and implement better resource allocation policies, are given. Experimental results of algorithm are presented in section 5. Section 6 contains conclusions and future work.

Figure 1 Consumers’ request, Revenue generation and deadline are interrelated to each other.
2. LITERATURE SURVEY

Cloud Computing offers leasing computing capacity on a pay per use basis based on service level agreement. It saves time and cost of installing hardware and software at consumers’ premises. Cloud provides services which are elastic in nature. [4], [5].

Computing infrastructure is provided to consumer in the form of virtual machines that can be used for different types of needs. IaaS service model provides tools and technologies in such a way that an organization’s existing hardware can be used to provide hardware services to other organizations or same organization. So an organization can use a cloud for its services or can become a cloud host and provide services to others [6].

In [7] provides an algorithm and a leasing policy for prioritizing consumers on the basis of CRI score. Authors showed policy to maintain order of execution of users’ task on the basis of some parameters and claimed their result is an improvement over existing policies and algorithms under some conditions.

Improvement on automated allocation and de-allocation of resources, resource monitoring, resource discovery and registration are provided in [8]. This work increases chance of adoption of clouds by the consumers. [8], [9].

Aneka’s deadline-driven provisioning mechanism is given in [10]. This mechanism is responsible for supporting quality of service (QoS)-aware execution of scientific applications in hybrid clouds. This mechanism can use and manage resources obtained from same or a variety of sources that include desktop grids, clusters, grids, public and private clouds.

Particle Swarm Optimization (PSO) is utilized to schedule workflows in the algorithm proposed in [11]. This was developed to work in clouds with a single-level SLAs and on-demand resource leasing. It does not consider workflow deadlines, but focuses only on monetary cost minimization.

IaaS cloud can be setup by using toolkits OpenNebula [12], Eucalyptus [13], OpenStack [14] etc.. Currently, OpenNebula scheduler provides an implementation of the Rank Scheduling Policy which aims to prioritize those resources more suitable for the VM. Match making scheduler of OpenNebula works on the match making algorithm. This algorithm works on policy from 0 to 4. 0th policy is packing, which Minimize the number of hosts in use by packing the VMs in the hosts to reduce VM fragmentation. 1st policy is stripping, which Maximizes resources available for the VMs by spreading the VMs in the hosts policies provided. 2nd policy is load-aware that maximize resources available for the VMs by using those nodes with less load. 3rd policy is Custom that uses a custom RANK. 4th policy is named Fixed in which, hosts will be ranked according to the PRIORITY attribute found in the Host or Cluster template.

Haizea, a toolkit developed in Python to work as a custom scheduler for opennebula or in simulation mode was introduced in [15]. New tailor made policies can be developed in Haizea. Haizea basically provides four types of leases: Best-effort (BE), Advanced reservation, Immediate leases and Deadline sensitive leases [16]. BE leases are preemptable and breakable i.e. resources can be given in different time intervals to every BE leases. BE leases does not have any time constraints, these leases are served as and when lease load permits the system. Immediate and advance reservation (AR) leases are non-preemptable and have time constraints, such as start time and end time. AR leases have more priority than BE leases for resources. It will preempt best-effort leases whenever the resources are required for advance reservation. There is no guarantee that a submitted best-effort lease will get resources for completion within a certain time limit [17].

If the system has lots of AR and immediate leases then priority will be given to these leases. BE leases have to wait for a long time and will be served when resources are free from AR & immediate leases [18]. To protect these leases from starvation, Starvation-removal algorithm was proposed in [19]. This algorithm supports very less suspensions of leases by counting number of suspensions and then increases priority of that lease by aging mechanism.

BE leases takes long period of time to complete due to presence of so many AR and immediate leases. This may lead to more requests to AR leases which in turn may cause internal fragmentation of free resources. Deadline sensitive (DLS) leases can also be used with Haizea. DLS are assumed to be preemptable but there is a limitation to their preemptability. It is preemptable only if the scheduling algorithm of Haizea can assure that it can be completed before its deadline. Thus, it will assure (1) to consumers that their request will be completed within a certain time limit (2) minimum internal fragmentation [20], [21], [22].

In [16], authors extended deadline sensitive algorithm of Haizea by considering several time slots that together can satisfy complete lease within given deadline. Leases with same resource request or if first lease request has fewer resources then second lease then only swapping of lease can be done to handle deadline situation. Other method proposed by the authors was backfilling but unfortunately, they did not implement it and left it as future work.

In [24] authors proposed Auto-scaling approach to minimize cost and meet application deadlines in cloud workflows. In this approach, the basic computing elements are virtual machines (VMs) of various sizes/costs, jobs. These VMs jobs are specified as workflows. Users specify performance requirements by assigning (soft) deadlines to jobs, and the goal is to ensure all jobs are finished within their deadlines at minimum financial cost. Authors proved their work by dynamically allocating/deallocating VMs and scheduling tasks on the most cost-efficient instances.

The deadline-driven cost-minimization algorithm [25], or the Deadline-Markov Decision Process (MDP), breaks the workflow into partitions. It assigns a maximum finishing time for each partition according to the deadline set by the user. Based on this time, each partition is scheduled for that resource which will result in the lowest cost and earliest estimated finishing time. This algorithm works with on-demand resource reservation.
Deadline and Budget distribution-based Cost-Time Optimization (DBD-CTO) workflow scheduling algorithm was proposed in [26]. This algorithm minimizes execution cost while meeting timeframe for delivering results, but this algorithm’s work with real time workflows left as future work. In our work, we are applying earliest deadline first algorithm [27] with a modification that, it should minimize the capacity required at cloud host side.

3. NEED OF mEDF ALGORITHM

Earliest deadline first (EDF) or least time to go is a dynamic scheduling algorithm. EDF is mostly used in real-time operating systems to place processes in a priority queue. Processes are arranged in queue according to their deadlines. Process queue is searched on every event of existing process finishes or a new process arrives for the process which is closest to its deadline. This process is the next to be scheduled for execution [28].

In IaaS cloud environment, lease requests arrivals are independent. These lease requests can be characterized by arrival time, duration and a deadline. Number of nodes required by the lease request can also be a factor that can play a vital role for optimizing computing capacity requirement at cloud host side in scheduling. Our mEDF algorithm can take these factors into account and can lead to improve waiting time and completion time of lease requests.

4. PROPOSED ALGORITHM

Algorithm required to implement mEDF policy are given in this section. Computing environment, test lease requests and algorithm are described in next subsections.

Computing Environment

A collection of 4 nodes, all with one CPU and 1024MB of memory are used as system for testing in simulated mode of Haizea tool kit.

```xml
<nodes>
  <node-set numnodes="4">
    <res type="CPU" amount="100"/>
    <res type="Memory" amount="1024"/>
  </node-set>
</nodes>
```

Sample Lease request

```xml
<lease-workload name="sample">
  <description>
    A simple trace where only one lease with deadline is specified is shown as sample here.
  </description>
  <lease-requests>
    <!-- First lease request -->
    <lease-request arrival="00:00:00">
      <lease preemptible="true">
        <nodes>
          <node-set numnodes="2">
            <res type="CPU" amount="100"/>
            <res type="Memory" amount="1024"/>
          </node-set>
        </nodes>
        <start/>
        <duration time="10:00:00"/>
        <deadline>
          <exact time="01-01-2014 00:00:00"/>
        </deadline>
        <software>
          <disk-image id="foobar.img" size="1024"/>
        </software>
      </lease>
    </lease-request>
  </lease-requests>
</lease-workload>
```
Attributes and their sub attributes like lease, lease id, nodes, start, duration time, and software are already available with Haizea installation. We have introduced new attribute deadline, so that a consumers’ tasks can be given a deadline. Monte Carlo simulation is used for generating durations and Poisson distribution is used for lease request arrival times for experiments.

Haizea may use backfilling algorithm with aggressive, conservative and off settings. We have used default aggressive settings for our experiments. Rejection of leases which have crossed deadline limit has also been developed. Lease requests arrive in their natural order. mEDF sorts all the lease requests in increasing order of deadline and number of nodes required for leases and enqueue them time to time taking care of backfilling process.

Algorithm

Start

For all leases i = 1 to n do

Begin

Enqueue request as arrived in queue
Sort leases according to Deadline, number of number of nodes
If Numnodes_requested <= Numnodes_available and deadline >= available_schedule then
Accept lease
Else
Begin
Reject lease
If deadline >= available_schedule then
Reschedule lease
Else
Delete lease entry from queue
End
If lease completed then
Delete lease entry from queue
End

Stop

5. EXPERIMENTS AND RESULTS

Table-1 and Table-2 shows consumers’ lease requests under test run with existing DLS and mEDF algorithm respectively. This batch of consumers’ lease requests is generated only for checking the validity of our proposed leasing policy and algorithm in lab. Parameters Lid, Req. Arrival, Duration in days, Deadline, NumNodes, Response time are used in table, where Lid is lease id usually given in ascending order in order of appearance of leases, Req. Arrival is date and time of consumers’ lease request, Duration in days is also a parameter which specifies requested amount of time for cloud services, NumNodes is number of nodes required by consumer in present lease, and Response time is time at which first response given to a particular lease and completion time is last date and time of providing nodes to consumers’ lease request.

Graph in figure-2 shows comparison between DLS algorithm and mEDF algorithm tested in Haizea tool kit.

Table -1: Deadline sensitive algorithm

<table>
<thead>
<tr>
<th>Lid</th>
<th>Req. Arrival</th>
<th>Duration in days</th>
<th>Deadline</th>
<th>Num Nodes</th>
<th>Response time</th>
<th>Completion time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01-01-2014 12:00 AM</td>
<td>10</td>
<td>14-01-2014 12:00 AM</td>
<td>2</td>
<td>01-01-2014</td>
<td>10-01-2014</td>
</tr>
<tr>
<td>2</td>
<td>01-01-2014 12:00 AM</td>
<td>8</td>
<td>12-01-2014 12:00 AM</td>
<td>2</td>
<td>01-01-2014</td>
<td>08-01-2014</td>
</tr>
<tr>
<td>3</td>
<td>01-01-2014 01:00 AM</td>
<td>5</td>
<td>09-01-2014 01:00 AM</td>
<td>2</td>
<td>Rejected</td>
<td>Rejected</td>
</tr>
<tr>
<td>4</td>
<td>02-01-2014 12:00 AM</td>
<td>15</td>
<td>16-02-2014 12:00 AM</td>
<td>4</td>
<td>08-Jan-2014</td>
<td>25-01-2014</td>
</tr>
<tr>
<td>5</td>
<td>03-01-2014 12:00 AM</td>
<td>12</td>
<td>19-02-2014 12:00 AM</td>
<td>4</td>
<td>16-Jan-2014</td>
<td>07-02-2014</td>
</tr>
<tr>
<td>6</td>
<td>04-01-2014 12:00 AM</td>
<td>4</td>
<td>07-02-2014 12:00 AM</td>
<td>1</td>
<td>29-Jan-2014</td>
<td>01-02-2014</td>
</tr>
<tr>
<td>7</td>
<td>05-01-2014 12:00 AM</td>
<td>10</td>
<td>18-01-2014 12:00 AM</td>
<td>3</td>
<td>Rejected</td>
<td>Rejected</td>
</tr>
<tr>
<td>8</td>
<td>05-01-2014 12:00 AM</td>
<td>3</td>
<td>17-02-2014 12:00 AM</td>
<td>2</td>
<td>02-Feb-2014</td>
<td>05-02-2014</td>
</tr>
</tbody>
</table>
Table 2: mEDF algorithm with same requests as in Table 1

<table>
<thead>
<tr>
<th>Lid</th>
<th>Req. Arrival</th>
<th>Duration in days</th>
<th>Deadline</th>
<th>Num Nodes</th>
<th>Response time</th>
<th>Completion time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01-01-2014 12:00 AM</td>
<td>10</td>
<td>14-01-2014 12:00 AM</td>
<td>2</td>
<td>Rejected</td>
<td>Rejected</td>
</tr>
<tr>
<td>2</td>
<td>01-01-2014 12:00 AM</td>
<td>8</td>
<td>12-01-2014 12:00 AM</td>
<td>2</td>
<td>01-01-2014</td>
<td>08-01-2014</td>
</tr>
<tr>
<td>3</td>
<td>01-01-2014 12:00 PM</td>
<td>5</td>
<td>09-01-2014 12:00 PM</td>
<td>2</td>
<td>01-01-2014</td>
<td>06-Jan-14</td>
</tr>
<tr>
<td>4</td>
<td>02-01-2014 12:00 AM</td>
<td>15</td>
<td>16-02-2014 12:00 AM</td>
<td>4</td>
<td>10-01-2014</td>
<td>26-Jan-14</td>
</tr>
<tr>
<td>5</td>
<td>03-01-2014 12:00 AM</td>
<td>12</td>
<td>19-02-2014 12:00 AM</td>
<td>4</td>
<td>16-01-2014</td>
<td>10-Feb-14</td>
</tr>
<tr>
<td>6</td>
<td>04-01-2014 12:00 AM</td>
<td>4</td>
<td>07-02-2014 12:00 AM</td>
<td>1</td>
<td>07-01-2014</td>
<td>11-Jan-14</td>
</tr>
<tr>
<td>7</td>
<td>05-01-2014 12:00 AM</td>
<td>10</td>
<td>18-01-2014 12:00 AM</td>
<td>3</td>
<td>Rejected</td>
<td>Rejected</td>
</tr>
<tr>
<td>8</td>
<td>05-01-2014 12:00 AM</td>
<td>3</td>
<td>17-02-2014 12:00 AM</td>
<td>2</td>
<td>27-01-2014</td>
<td>30-01-2014</td>
</tr>
<tr>
<td>9</td>
<td>07-01-2014 12:00 AM</td>
<td>10</td>
<td>08-03-2014 12:00 AM</td>
<td>2</td>
<td>01-02-2014</td>
<td>10-02-2014</td>
</tr>
<tr>
<td>10</td>
<td>08-01-2014 12:00 AM</td>
<td>7</td>
<td>16-03-2014 12:00 AM</td>
<td>2</td>
<td>26-01-2014</td>
<td>02-02-2014</td>
</tr>
</tbody>
</table>
6. CONCLUSION AND FUTURE WORK

mEDF algorithm implementation shows improvement in response time and completion time of various leases. This algorithm can be very beneficial to cloud hosts and consumers. Proposed algorithm will automatically reject the leases, which cannot be completed in time thus it saves resources for other leases and improves response time and completion time of lease requests of consumers. Leases with scientific workloads will get most benefitted by this algorithm. Future work may use experimenting same algorithm with backfilling algorithm’s conservative and off modes and provide a comparative study of all deadline based algorithms. This work also suggests taking CRI score [7] along with mEDF algorithm to improve resource utilization and revenue at cloud host.

References


AUTHORS

Vivek Shrivastava is pursuing Ph. D. in CS from DAVV under the guidance of Dr. D.S. Bhilare. He is M.Tech. (Computer Sc.), MCA, and UGC-NET qualified in Computer Sc. & Applications. Working as an Assistant Professor in International Institute of Professional Studies, Devi Ahilya University, Indore. His areas of interest are Cloud Computing, Ubiquitous Computing and Information Security.

D.S. Bhilare received his Ph.D. (Computer Science), M.Tech. (Computer Sc.), M.Phil. (Computer Sc.) and MBA from Devi Ahilya University, Indore. Worked as a senior project leader for ten years in the industry and developed various business applications for different industries. Since last twenty years, working in the University as a Senior Manager & Head Computer Centre. He has presented several papers in national and international conferences and published in reputed journals and international conference proceedings in the area of Information Security.