ISSN:0975-9646

CRI: A Novel Rating Based Leasing Policy and Algorithm for Efficient Resource Management in IaaS Clouds

Vivek Shrivastava[#], D. S. Bhilare[*]

[#]*International Institute of Professional Studies, Devi Ahilya University, Indore, India*

[*]*Computer Centre, Devi Ahilya University, Indore, India*

***Abstract*— Cloud Computing is transfiguring development of information technology industry by providing scalable services on a pay per use basis. Cloud hosts and consumers are tied with service level agreement (SLA). SLA provides description of services provided by the cloud host. A cloud host can serve to multiple consumers and this is called cloud computing multi tenant model. Though, cloud shows infinite capacity, still there must be some policy to maintain order of execution of users' tasks. Consumers can be served in a first come first serve, round robin or on a priority basis. This paper proposes a new leasing policy named CRI (Consumer Rating Index) and an algorithm for prioritizing consumers on the basis of CRI scores. Experimental results show that this policy and algorithm can be used for efficient functioning at cloud hosts side.**

Keywords— **IaaS, Resource management, Rating based priority, CRI.**

I. Introduction

Cloud computing is provisioning elastic computing services on demand as a utility. Consumers are billed only for what they availed by the host. 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).

Consumers are free from one time heavy monetary investment in terms of software, platform, and computing infrastructure. Instead consumers can avail services on lease basis, and can change to a different cloud host after using services of one due to (1) more promising features of other cloud hosts, (2) less satisfaction of services from its current cloud host. SLA is used as a descriptive document between both parties: cloud host and consumer.

Resource management is main challenging area in cloud computing. A resource manager in cloud computing is responsible for managing the resources available. Resources may include hardware such as the node, the CPUs, memory, network bandwidth or software available on certain nodes [1],

[2]. Resource manager also defines the queues and handles job start up and shutdown, but does not determine which jobs to start or stop. Resource management affects performance, functionality and cost of a system. Especially in IaaS clouds, resources must be used in a cost-performance ratio. Various different types of algorithms are used in cloud to maximize the usage of resources in a cost effective manner.

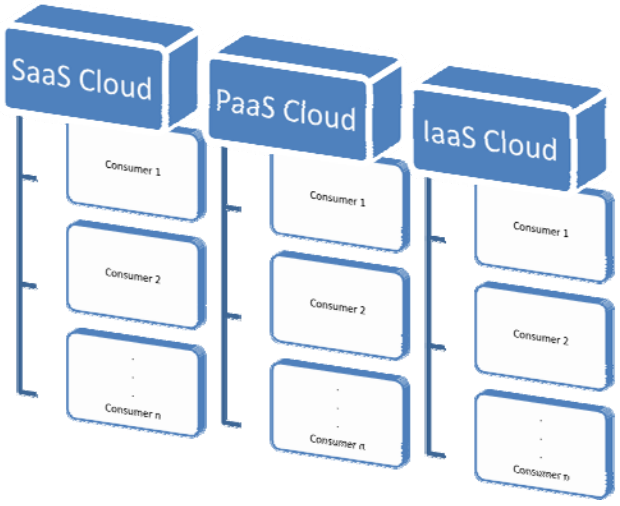

Fig. 1 Cloud computing multi tenant model

Resource scheduling is key aspect of resource management. The cloud scheduler examines the running jobs and jobs in ready queue. Different types of scheduling policies may be available for scheduler. Cloud scheduler applies proper scheduling policy to schedule resources to jobs and keeps track of resource usage and free available resources [1], [3], [4]. Scheduler instructs resource manager to start or stop the job on specific node, based on scheduling policy like FCFS, Round Robin, mDelay [5] etc. . Cloud resource scheduler plays a resource monitor role. Cloud resource scheduler is also in-charge of reservations of resources for a particular time.

Scheduling algorithms can be divided into different categories like: static scheduling and dynamic scheduling algorithms, centralized hierarchic and distributed scheduling algorithms, pre-emptive and non pre-emptive algorithms, immediate and batch mode scheduling algorithms, and independent and workflow scheduling algorithms. Classification of scheduling algorithm can also be done according to objective functions like: Application centric (Make-span, economic cost algorithms), Resource centric (resource utilization, economic profit) and on the basis of providers [6].

Computing infrastructure is provided in terms of virtual machines (VMs) on a lease basis to the consumers. In batch mode scheduling, best effort leases can be used. Consumers can be categorized according to the CRI score.

Rest of the paper is organized as: Section 2 contains related work. Section 3 describes proposed leasing policy CRI for handling different consumers' task. In section 4, proposed algorithm, to serve consumers in order of their CRI score and implement better resource allocation policies, are given. Experimental results of algorithm and graph are presented in section 5. Section 6 contains conclusions and future work.

II. 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 [7], [8], [9].

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 its own private cloud for its services or can become a public cloud host and provide services to others [10].

Service Level Agreement (SLA) is used as a contract between cloud host and consumer. This SLA may provide conditions that must be followed during operation like: quality of services (QoS), time at which services will be available, soft deadline or hard deadline and many more other conditions. SLA monitoring and SLA enforcement tools are also available and trust issues can be solved using Web Service Level Agreement (WSLA) as proposed in [11].

Resource management is required to keep SLA. Mehta et al. proposed automated resource management. Automated resource management enables cloud hosts to adopt quickly changes in service demands of consumers [12], [13]. Resource allocation and de-allocation requires some policy so that consumers must not suffer unavailability of resources for a long period of time.

Resource management and scheduling provided by OpenNebula are benefiting to most cloud users that are using it. OpenNebula is a well known toolkit for setting up an IaaS cloud. Currently, OpenNebula scheduler provides an implementation of the Rank Scheduling Policy. This Rank Scheduling Policy aims to prioritize resources that are more suitable for the VM. Match making scheduler of OpenNebula as described in [14], works on the match making algorithm which works on policies provided in table 1.

TABLE I ,Open Nebula's Match Making Algorithm Policies [14]

Policy	Description
0	Packing: Minimize the number of hosts in use by packing the VMs in the hosts to reduce VM fragmentation.
1	Striping: Maximize resources available for the VMs by spreading the VMs in the hosts.
2	Load-aware: Maximize resources available for the VMs by using those nodes with fewer loads.
3	Custom: Use a custom RANK.
4	Fixed: Hosts will be ranked according to the PRIORITY attribute found in the Host or Cluster template.

Haizea can be used as a custom scheduler for OpenNebula [15]. New policies can also be implemented in Haizea [16]. Haizea lease manager provides four types of leases: Immediate, Best Effort (BE), Advance Reservation (AR) and Dead Line Sensitive (DLS) [17].

A set of Best Effort (BE) leases can go indefinite number of postponement due to presence of Advance Reservation (AR) leases. BE leases can be protected from starvation by adopting starvation-removal technique introduced in [18]. Efficient resource utilization and scheduling can also be done using CBUD Micro, introduced in [8]. CBUD Micro can be used to measure performance of machines used at cloud host side and consumer side.

Scheduling can be done in terms of priority of some attribute. A high priority determines more important and low priority leads to less importance. In terms of task scheduling, it determines the order of task scheduling based on the parameters undertaken for its computation [19]. Work done in [20], is to maximize profit by executing tasks with higher profit on minimum cost based machine. The tasks are prioritized on the basis of task deadline in ascending order and task profit in descending order. We are proposing a new leasing policy to prioritize consumers' lease requests according to a ten-tuple as described in subsequent sections.

III. CRI Leasing Policy

In our work, we are proposing CRI leasing policy. Different consumers' request for lease can be given priority according to CRI score. CRI can be characterized by following set of parameters:

CRI =<H, TO, POW, CB, TOC, LTT, SV, EOW, FR, NFR > (3.1)

These parameters can be rated from 1-10 scale by cloud host.

Here H	=History of consumer with cloud host. For example a consumer can be a new consumer or an old consumer.
TO	=TurnOver of consumer, turnover may be helpful in deciding financial position of consumer.
POW	=Purpose Of Work. This work may be of type scientific (suitable for grids) or commercial (suitable for clouds).
CB	=Current Balance with cloud host, to decide consumers lease limit with cloud host.
TOC	=Type Of Consumer, where a consumer can be an individual, a private company, a government or public organization.
LTT	=Lease Total Time, so that resources can be reserved or relinquished for other consumers.
SV	=Security Vulnerability, Different groups may be rival to each other or some consumers may have wrong intentions with other consumers. If a consumer has low reputation in terms of security, his overall CRI may suffer.
EOW	=Essentiality Of Work. This EOW may be given by the consumers by mentioning degree of essentiality of their work.
FR	=Functional Requirements of consumers.
NFR	=Non-Functional requirements of consumers.

CRI score for a consumer can be calculated as:

$$CRI = \frac{H*w_1+TO*w_2+POW*w_3+CB*w_4+TOC*w_5+LTT*w_6+SV*w_7+EOW*w_8+FR*w_9+NFR*w_{10}}{\sum_{i=1}^{10} wi} \qquad (3.2)$$

Where w_1 to w_{10} are weights of aforesaid parameters and these weights can vary across different cloud hosts and may be proprietary to cloud host.

This leasing policy provides a common basis for evaluating different consumers. Further this CRI can be calculated by organization as according to its needs and application area. According to the CRI leasing policy, we can infer four important corollaries that will be used in the algorithm.

Corollary 1. *CRI score may be higher or lower of different consumers. A Consumer having low CRI will be served first.*

This can be easily inferred from the specification of the CRI leasing policy. A cloud host According to CRI score will provide VMs to different consumers. High CRI score will lead to delay in provisioning VMs to a consumer, while a low CRI score will provide a consumer early response for VMs.

Corollary 2. *All leases to be served can be executed in batch mode. Algorithm is most suitable for scientific type of computations.*

Consumers are sequenced according to CRI score, so immediate lease requirement or advance reservation (AR) lease are not suitable for proposed algorithms. A homogeneous lease requirement is constraint of this leasing policy. Scientific computations may be batched for such situations.

Corollary 3. *If two or more consumers (say Ai, Ai+1, Ai+2,…..Ai+n, for 1<=i <= k) have same CRI score, then consumer Ai will be given VMs prior than Ai+1 if Ai requires less number of VMs than Ai+1, for 1 <= i <= k-n).*

Consumers having same CRI score will be given VMs one after another according to less number of leases required by consumer.

Corollary 4. *If two or more consumers (say Ai, Ai+1, Ai+2, .., Ai+n, for 1<=i <= k) have same CRI and number of VMs required, then consumers will be served in a first come first serve manner.*

Consumers having same CRI score and same number of VMs requirement will be leased in order of their requests to cloud host.

IV. Proposed Algorithms

Algorithm based on proposed leasing policy CRI is given in this section. System description and Algorithm are also described here. Sample lease requests are described in later and computing environment is described in former paragraph.

A. System Description

In Haizea, input of leases can be done with the help of XML based files called lease workload file (lwf), further applications can enter lease request in XML file and then this XML files can be given as input to Haizea lease manager. The following specifies a collection of 12 nodes, all with one CPU, four with 1024MB of memory and eight with 2048MB of memory as used in [18]:

```
<nodes>
        <node-set numnodes="4">
        <res type="CPU" amount="100"/>
        <res type="Memory" amount="1024"/>
        </node-set>
        <node-set numnodes="8">
        <res type="CPU" amount="100"/>
        <res type="Memory" amount="2048"/>
        </node-set>
</nodes>
```

Sample lease request in modified lwf is shown below:

```
<lease-workload name="sample">
        <description>
                A simple trace where so many
          leases with CRI as rating is specified.
        </description>
<lease-requests>
<!-- First lease request-->
        <lease-request arrival="00:00:00">
        <lease preemptible="true">
        <nodes>
                <node-set numnodes="4">
                <res type="CPU"
                amount="100"/>
                <res type="Memory"
                amount="1024"/>
                </node-set>
        </nodes>
        <start></start>
        <duration time="10:00:00"/>
        <software>
                <disk-image id="foobar.img"
             size="1024"/>
        </software>
        <rating  rat="4"/>
</lease>

</lease-request>
<!-- Second lease request -->
<lease-request arrival="01:00:00">
        <lease preemptible="true">
        <nodes>

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

```
                </node-set>
            </nodes>
            <start></start>
            <duration time="10:00:00"/>
            <software>
                <disk-image id="foobar.img"
              size="1024"/>
            </software>
            <rating  rat="2"/>
            </lease>
    </lease-request>
    <!-- Third lease request -->
    <lease-request arrival="02:00:00">
            <lease preemptible="true">
            <nodes>
                <node-set numnodes="4">
                <res type="CPU"
             amount="100"/>
                <res type="Memory"
              amount="1024"/>
            </node-set>
            </nodes>
            <start></start>
            <duration time="10:00:00"/>
            <software>
                <disk-image id="foobar.img"
             size="1024"/>
            </software>
            <rating  rat="1"/>
            </lease>
    </lease-request>
    <!-- Fourth lease request -->
    <lease-request arrival="03:00:00">
            <lease preemptible="true">
            <nodes>
                <node-set numnodes="4">
                <res type="CPU"
             amount="100"/>
                <res type="Memory"
              amount="1024"/>
                </node-set>
            </nodes>
            <start></start>
            <duration time="10:00:00"/>
            <software>
                <disk-image id="foobar.img"
              size="1024"/>
            </software>
            <rating  rat="3"/>
            </lease>
    </lease-request>
```

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 rating, which represents CRI score of consumer. This CRI score will be calculated by cloud host with the help of equation (3.2), according to its needs, application area and present situations.

Haizea is implemented in Python language, so we also have implemented our algorithms in Python and integrate those modules with Haizea to test and run. Procedure Enqueue_Lease is used to append a new request to lease queue. We have modified BE leases according to our need to implement CRI leases so, we are interested in only BE lease environment. Procedure Sort_Queue is used to sort all available leases in using CRI score as primary key and number of VMs as secondary key. Serve_Leases procedure is used to do operations of Haizea normally i.e. managing available lease requests.

B. Algorithm

```
Procedure Enqueue_Lease()
Begin
    if scheduling lease type = BE then:
        insert_in_queue(lease, CRI,
         Number_of_nodes)
        call Sort_Queue (lease,
    Number_of_nodes)
        call Sort_Queue (lease, CRI)
    end if
End of Enqueue_Lease()

Procedure Sort_Queue(lease, CRI)
Begin
    Sort all leases according to Timsort Algorithm [21]
    call Serve_sorted_leases()
End of  Sort_Queue(lease, CRI)

Procedure Serve_Leases()
Begin
    if resources are available then:
        Pick leases from sorted queue and allot
    VMs demanded.
        if VM_shutdown=true then:
            Relinquish VMs for other leases.
        else
            put leases in wait queue
        end if
    end if
End of Serve_Leases()
```

V. Experiments and Results

Table-1 shows consumers' requests under test run. This batch of consumers' requests is generated only for checking the validity of our proposed leasing policy and algorithm in lab. Parameters Lid, CRI, NumNodes, C_No are used in table, where Lid is lease id usually given in ascending order in order of appearance of leases, CRI score is consumer rating index score, NumNodes is number of nodes required by consumer in present lease, and C_No is completion number of that lease.

Graph in figure-2 shows comparison between existing algorithm and proposed algorithm implemented by us in Haizea.

TABLE II.
RESULTS SHOWING COMPLETION OF LEASES ACCORDING TO CRI SCORE AS PRIMARY KEY AND NUM_NODES AS SECONDARY KEY.

Sr. No.	Lid	CRI Score	Num_Nodes	C_No
1	1	4	4	5
2	2	3	2	4
3	3	5	2	6
4	4	1	4	1
5	5	2	4	3
6	6	2	2	2
7	7	6	1	7

VI. CONCLUSION AND FUTURE WORK

CRI leasing policy and its implementation can be very beneficial to both cloud hosts and consumers. This leasing policy is flexible and provides choice of weight to cloud host. A large number of requests of leases by different consumers can be easily served by using our leasing policy. At the time of peak load this leasing policy provides a basis to accept or reject consumers' request according to its CRI score. In case of same CRI, consumers requests are served according to low-to-high number of VMs required so that a small task can be finished first. If consumers have same CRI score and same number of VMs i.e. same primary and secondary key for sorting then their order of arrival of lease request's date and time will be used as third key to sort such lease requests.

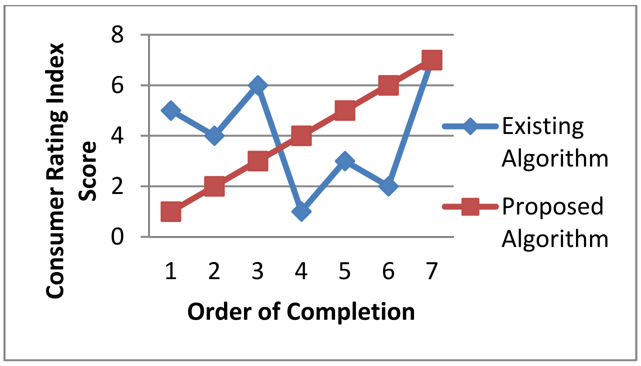

Fig. 2 Graph showing comparisons between existing and proposed algorithms.

Future work may use prioritizing CRI as primary key and less number of resources required for small amount of time as secondary key. This work also suggests a comparative study of serving consumers' requests as primary key and sorting number of nodes in descending order as secondary key to see the overall effect on cost and revenue at cloud host side.

REFERENCES

[1] *P. Brucker, Scheduling algorithms,* 5th ed, *Berlin*: *Springer-*Verlag, *2007.*

[2] Ti Leggett. (2012) HPC job management page on wiki. [Online]. Available: http://wiki.ci.uchicago.edu/Resources/JobManagement.

[3] D. Poola, S. K. Garg, R. Buyya, Y. Yang, and R. Kotagiri. "Robust scheduling of scientific workflows with deadline and budget constraints in clouds," In Proc. *The 28th IEEE AINA-2014*, 2014, paper p. 1-8.

[4] H. K. Mehta, P. Kanungo, and M. Chandwani. "Performance enhancement of scheduling algorithms in clusters and grids using improved dynamic load balancing techniques," In Proc. *The 20th ACM ICCWWW-2011*, 2011, p. 385-390.

[5] H. K. Mehta, M. Chandwani, and P. Kanungo. "A modified delay strategy for dynamic load balancing in cluster and grid environment," In Proc. *IEEE ICISA*, 2010, paper p. 1-8.

[6] Michael Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 2nd Ed., NJ: Prentice-Hall, 2002.

[7] H. K. Mehta, and E. Gupta. "Economy Based Resource Allocation in IaaS Cloud," *International Journal of Cloud Applications and Computing* vol. 3, (2), pp. 1-11, 2013.

[8] V. Shrivastava, and D. S. Bhilare. "CBUD Micro: A Micro Benchmark for Performance Measurement and Resource Management in IaaS Clouds,"
International Journal of Emerging Technolgy and Advanced Engineering vol. 3(11), pp. 433-437, 2013.

[9] S. Ostermann, I. Alexandria, Y. Nezih, P. Radu, F. Thomas, and E. Dick. "A performance analysis of EC2 cloud computing services for scientific computing," In Proc. ICCC, 2010, paper, p. 115-131.

[10] B. Sotomayor, S.M. Rubén, M. L. Ignacio, and I. Foster. "Virtual infrastructure management in private and hybrid clouds," *Internet Computing, IEEE* vol. 13(5), pp. 14-22, 2009.

[11] P. Patel, A. Ranabahu, and A. Sheth. "Service level agreement in cloud computing." Wright state University, Kno.e.sis Tech. Rep., 2009.

[12] H. Mehta, P. Kanungo, and M. Chandwani. "Performance enhancement of scheduling algorithms in web server clusters using improved dynamic load balancing policies," in Proc. *INDIACom-2008,p. 651-656* .

[13] Q. Zhang, C. Lu, and B. Raouf. "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications* vol 1(1), pp. 7-18, 2010.

[14] (2014) The OpenNebula Website. [Online]. Available: http://opennebula.org/.

[15] (2014) The Haizea Website. [Online]. Avaialable: http://haizea.cs.uchicago.edu/.

[16] B. Sotomayor. (2009) The Haizea Manual on Haizea. [Online]. Available: *http://haizea.cs.uchicago.edu/haizea_manual.pdf.*

[17] A. Nathani, S. Chaudhary, and G. Somani. "Policy based resource allocation in IaaS cloud," *Future Generation Computer Systems* Vol. 28(1) pp. 94-103, 2012.

[18] V. Shrivastava, and D. S. Bhilare. "Algorithms to Improve Resource Utilization and Request Acceptance Rate in IaaS Cloud Scheduling," *International Journal of Advanced Networking & Applications* vol. 3(5), pp. 1367-1374, 2012.

[19] W. Lin, L. Chen, Z. W. James, and R. Buyya. "Bandwidth-aware divisible task scheduling for cloud computing," *Software: Practice and Experience* vol 44(2), pp. 163-174, 2014.

[20] M. Choudhary, and S. K. Peddoju. "A dynamic optimization algorithm for task scheduling in cloud environment," *International Journal of Engineering Research and Applications* vol. 2(3), pp. 2564-2568, 2012.

[21] T. Peters. (2002), listsort page on svn.python [Online]. Available: http://svn.python.org/projects/python/trunk/Objects/listsort.txt