Int. J. Advanced Networking and Applications
Volume: 03, Issue: 05, Pages: 1367-1374 (2012)

Algorithms to Improve Resource Utilization and Request Acceptance Rate in IaaS Cloud Scheduling

Vivek Shrivastava
International Institute of Professional Studies, Devi Ahilya University, Indore-17
Email: shrivastava.vivex@gmail.com
D.S. Bhilare
Computer Centre, Devi Ahilya University, Indore-17
Email: bhilare@hotmail.com

----------*ABSTRACT*----------

Computing infrastructure provisioning in Infrastructure as a Service (IaaS) cloud is done in the form of virtual machines. Haizea, a resource lease manager, provides four types of leases: Immediate, Best Effort (BE), Advance Reservation (AR) and Dead Line Sensitive (DLS). AR leases are most privileged leases with "AR preempts other leases" policy, since they can preempt & suspend other BE leases when demanded by consumers. This leads to two problems: 1) a set of BEs can go suspended infinite number of time & 2) ARs, at the actual time of their resource allocation, can be rejected due to presence of other ARs in schedule. This work proposes two algorithms 1) Starvation-Removal and 2) AR-to-BE Conversion to solve these problems. Experimental results of the proposed algorithms successfully demonstrate that we can stop starvation of BE leases for resources and effectively improve request acceptance rate.

Keywords: **AR-to-BE Conversion, IaaS cloud, Haizea, Scheduling, Starvation-Removal.**

Date of Submission: September 01, 2011 Date of Acceptance: November 11, 2011

1. Introduction

Cloud Computing provisions the supply of computing resources on the basis of demand, as and when needed. Infrastructure as a Service (IaaS) is a model of cloud computing which provides infrastructure to different organizations according to their needs [1]. OpenNebula an open source tool supports IaaS model by managing virtual machines (VMs), as an abstraction of infrastructure, for private, public and hybrid clouds [2], [3]. OpenNebula uses only Immediate lease provisioning to schedule IaaS cloud resources using Match-making Algorithm. The Match-making algorithm as described in [4] allocates resources with a higher RANK expression first to allocate VMs. This RANK expression is important in applying placement policies like Packing, Striping and Load-aware policy. Packing policy minimizes the number of cluster nodes in use by using those nodes with more VMs running first. Striping policy maximizes the resources available to VMs in a node by using those nodes with less VMs running first while Load-aware policy do the same job by using those nodes with more free CPU first.

Resources available to OpenNebula can also be scheduled by replacing OpenNebula's native scheduler with Haizea, a lease manager, which is also an open source tool [5]. Haizea provides four types of leases which can be used to provide virtual machine services to clients. These four types of leases are (1) Immediate (2) Best Effort (BE) (3) Advance Reservation (AR) and (4) Dead Line Sensitive (DLS) [5], [6], [7].

Immediate lease is useful for the situations, when a request arises for resources & it needs immediate attention. If the resources are not available then this lease request is canceled by resource scheduler. In case of a resource request that do not require immediate attention and can wait for availability of resources BE lease is suitable. In BE lease policy- leases are queued up and processed in a first in first out manner on availability of resources. Whenever a new deadline sensitive lease is submitted to Haizea, scheduler tries to find a single time slot to satisfy the whole lease. If it is not able to find such a slot then it reschedules already accommodated deadline sensitive leases to make space for new lease. To do this, scheduler sorts the deadline sensitive leases based on slack value and reschedules them considering preemption. In [7] this behavior is changed by considering several free time slots available, which together can satisfy the complete lease within its deadline. This way several new leases may get scheduled without requiring rescheduling of other

Volume: 03, Issue: 05, Pages: 1367-1374 (2012)

leases to make space for them. AR lease request is done by the consumers or users when they need to use infrastructure in a time critical manner i.e. for fixed start and end time of lease. This AR lease can cause an already running BE lease scheduled on some nodes to suspend, so that resources provisioned to that lease can be preempt from the BE lease and can be provided to AR lease [5], [7]. Successive demands of AR lease, which suspends BE lease again and again can cause delay in completion of BE lease, thus can decrease interest of users or consumers in demanding BE lease. A large number of requests for AR lease can create bottleneck at IaaS provider side for resources. Presented work deals with this problem and proposes required algorithms and actions to be taken for increasing resource utilization by decreasing request rejection rate and protect BE lease from starvation of resources, & thus reduce overheads in completing BE leases.

Section 2 contains related work. Section 3 describes need of introducing Starvation-Removal and AR-to-BE Conversion algorithms. In section 4, proposed two algorithms, Starvation-Removal algorithm to prevent a BE for indefinite postponement and AR-to-BE Conversion algorithm to implement better resource allocation policies, are given. Experimental results of both algorithms are presented in section 5. Section 6 contains conclusions and future work.

2. Related Work

The Various research groups are exploring the ways to use cloud computing and IaaS cloud as next generations paradigm shift. Resource scheduling is most promising topic in IaaS cloud computing. Salehi et *al.* proposed economy driven resource management architecture for global computational grids [8]. This consists of a generic framework, called GRid Architecture for Computational Economy (GRACE). GRACE is intended for negotiation and trading resources dynamically in association with local schedulers and grid or meta-schedulers. Gridbus, in place of GRACE is used now by the authors [9].

Mehta et *al.* developed the web service based approach for resource registration and discovery (RRD) in distributed computing environment [10]. The service practiced by authors can be used in cloud computing environment to provide facility to discover the suitable resources, matching it with the processes and booking of the most appropriate resource. But that work does not address either the problem of starvation of resources of some lease requests, due to overuse of one type of resources at a point of time.

There has been some work on Grid-based environments with well defined priority based scheduling algorithms but applicability, adaptability and enhancement of these algorithms in IaaS cloud computing environment are to be tested [11].

Sotomayor et *al.* developed and explored Haizea's capacity for providing IaaS in [12], [13], [14], [15], and [16]. Authors had done a great work in the field of provisioning IaaS resources and had done a number of successful experiments for scheduling resources to the consumer at the required time of their requests.

Afoulki et *al.* suggested security aware scheduler for IaaS cloud computing because most IaaS cloud environment provides multi-tenant facility that is more than one user can use VM resources at same time from one source cloud. This may lead to cross VM attacks. Isolation of data and services are required for less trusted customers, which can steal data or services from other VMs [17].

Nathani et *al*. recommended deadline sensitive resource scheduling in IaaS cloud computing by experimenting in Haizea while minimizing the total number of leases rejected by it. Dynamic planning based scheduling algorithm has been implemented in Haizea by the authors, which can admit new leases and prepare the schedule whenever a new lease can be accommodated [7].

Akhani et *al*. proposed decision making algorithms and extended the current AR algorithms in IaaS cloud computing by experimenting in Haizea to provide negotiation based allocation of resources and they have done very well in dynamic negotiation of resources but proposed work in this paper does not focus this problem [6].

Cloud providers (e.g. Amazon EC2, Google Apps and Mosso) provide different kinds of schemes for resource allocation [6]. These schemes are described in Service level Agreement (SLA) format. Based on those SLAs, consumers can accept or reject the scheme [18]. Consumers can select the scheme provided by these commercial cloud providers but they cannot have information about any suspension times when they are using some BE kind of leasing scheme or rejection of lease at the time of lease-execution. By applying Starvation-Removal and AR-to-BE Conversion algorithms, proposed in this paper, process of provisioning resources becomes more probable to finish under an estimated time frame.

In [19] aging is described as the process of gradually increasing the priority of a task, based on its waiting time. The aging technique estimates the time a process will run based on a weighted average of previous estimates and measured values. Aging can be used to reduce starvation of low priority tasks. Similarly in context of BE lease suspensions, Starvation-Removal algorithm can be used as a factor which describes maximum limit for suspensions of a BE lease. Some leases can be prevented from starvation during their execution time and authors of this paper have used same approach to increase resource utilization and decrease request rejection rate.

3. Need of Starvation-Removal and AR-to-BE Conversion Algorithms

IaaS model of cloud computing can be seen as a revenue increasing conduct at IaaS provider side by providing various types of infrastructure differing in terms of processing power, memory & storage space. Different types of lease are used to provide IaaS. Any IaaS provider, to increase attraction of its consumers, can give so many choices and tries to increase his revenues by presenting all types of services on his menu. If all the consumers try for only one type of lease then that can create bottleneck at IaaS provider side and load for a particular lease will increase, this scene can be worse at peak service load time when there is a sudden increase in request rate of resources, due to which rejection rate can be increased.

This work identifies existing resource allocation policies provided by Haizea and finds a problem of increasing one type of lease policy requisition. This work proposes to increase use of all types of lease. IaaS cloud consumers may demand only AR policy instead of using other policies due to availability of heavy budget. If time critical applications are requiring provisioning of infrastructure resources then in that cases AR policy is most suitable, but when applications are not time critical, then they can be queued and provisioned resources by using BE lease in system until all resources required are not available. A BE lease, due to presence of so many successive AR leases can be postponed for infinite times. Every suspension of a BE lease requires suspension time and resumption time as overhead. So time required to complete any BE lease is addition of actual lease provisioned time, time required to suspension, time required to resumption, time spent in queue, and time increased due to bad bandwidth[12]. This can be explained as formula:-

BE Lease Completion time = lease execution time + LST + LRT + time spent in queue + time increased due to bad bandwidth (1)

where

$$LST = \sum_{i=1}^{n} LeaseSuspensionTime \quad (2)$$

and

$$LRT = \sum_{i=1}^{n} LeaseResumptionTime \quad (3)$$

here n is total number of suspension & resumption of that BE lease. As n grows large, time require to finish a BE lease also increases. Starvation-Removal algorithm presented in paper successfully overcomes this problem and increases chances of BE lease to finish in a small time interval by limiting number of suspensions.

Second problem is with AR leases. These type of leases require to request for resources before some substantial amount of time, but this does not guarantee that resources will be available to that AR lease in proper time-frame, and so, that AR lease can be rejected at exact time of starting lease-execution due to lack of resources at that time. This type of situations increase request rejection rate and thus degrade overall performance of IaaS cloud. AR-to-BE Conversion algorithm presented in this paper deals with this situation in an efficient and user-friendly manner and thus provides proper usage of resources and increase the efficiency of resource provisioning manager by reducing request rejection rate.

Proposed solutions presented in section 4 achieve flexible management of virtual machines, increase resource utilization, provide more options to choose suspension and rejection decisions to deploy VMs and BE lease using proposed algorithms so that IaaS model can be used more effectively. Proposed solutions schedules efficiently heterogeneous workloads (combining BE and AR leases), by overcoming the resource utilization problems.

4. Proposed Algorithms

Starvation-Removal algorithm prevents BE leases from starvation of resources by denying new AR leases to schedule after a fixed number of suspension of BE leases. AR-to-BE Conversion algorithm prevents AR lease rejection by converting AR lease to a BE lease, and thus queues lease to be rejected to run in future time frames.

Volume: 03, Issue: 05, Pages: 1367-1374 (2012)

4.1 Starvation-Removal Algorithm to prevent BE Leases from starvation of resources

Proposed algorithm changes the way of handling BE leases of existing algorithm as shown in figure 1. Capacity of system & lease request input is done by XML files shown in section 4.1.1.

4.1.1 System Description

In Haizea, input of leases can be done with the help of XML based files, 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.

```
<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>
```

AR leases can be identified as lease preemptible= false and exact time is given which schedules start time of providing resources. BE lease can be identified as lease preemptible= true and <start> </start> tag contains nothing while immediate lease can be identified as <start> <now> </start> and DLS lease can be identified as <deadline time="Time Value"/>. Format for specifying lease requests in Haizea can be shown as below:

```
<lease-workload name="sample">
   <description>
   A simple trace where an AR lease preempts a
   best-effort lease that is already running.
   </description>
   <lease-requests>
   <!-- First lease request   BE lease request-->
   <lease-request arrival="00:00:00">
   <lease preemptible="true">
   <nodes>
          <node-set numnodes="1">
                 <res type="CPU" amount="100"/>
                 <res type="Memory" amount="1024"/>
          </node-set>
   </nodes>
   <start></start>
   <duration time="01:00:00"/>
   <software>
   <disk-image id="foobar.img" size="1024"/>
   </software>
   </lease>
   </lease-request>
   <!-- Second lease request  AR lease request-->
   <lease-request arrival="00:15:00">
   <lease preemptible="false">
   <nodes>
          <node-set numnodes="4">
                 <res type="CPU" amount="100"/>
                 <res type="Memory" amount="1024"/>
          </node-set>
   </nodes>
   <start>  <exact time="00:30:00"/> </start>
   <duration time="00:30:00"/>
   <software>
   <disk-image id="foobar.img" size="1024"/>
   </software>
   </lease>
   </lease-request>
   <!-- Third lease request  Immediate lease request-->
   <lease-request arrival="00:15:00">
   <lease preemptible="true">
   <nodes>
          <node-set numnodes="1">
                 <res type="CPU" amount="50"/>
                 <res type="Memory" amount="1024"/>
          </node-set>
   </nodes>
   <start>  <now > </start>
   <duration time="00:15:00"/>
   <software>
   <disk-image id="foobar.img" size="1024"/>
   </software>
   </lease>
   </lease-request>
<!-- Fourth lease request DLS request -->
<lease-request arrival="00:20:00">
   <lease  preemptible="true">
      <nodes>
        <node-set numnodes="1">
          <res amount="100" type="CPU"/>
          <res amount="1024" type="Memory"/>
        </node-set>
</nodes>
   <start> <exact time="00:40:00.00"/> </start>
   <duration time="00:10:00.00"/>
    <deadline time="02:00:00.00"/>
    <software>
   <disk-image id="foobar1.img" size="1024"/>
   </software>
   </lease>
   </lease-request>
```

Existing algorithm is implemented in Python language, so we also have implemented our algorithms in Python & integrated that modules with Haizea to test and run.

Algorithm presented in figure 1 counts total number of suspensions of each & every BE lease independently. If any BE lease is suspended more than limit entered by user

then new AR leases are not be scheduled until that BE is finished. To implement this algorithm we have used a database in sqlite3 that is responsible for accountability of maximum suspensions of a particular BE. For each BE, maximum suspension limit can be entered by requester of that BE in advance, for our experimental purposes we specified BE maximum suspension limit equal to 2.

4.2 AR-to-BE Conversion Algorithm to prevent AR lease to be rejected

AR-to-BE Conversion algorithm is presented in figure 2. Our algorithm increases request acceptance rate by converting rejected AR leases to BE leases at dynamic time of their rejections.

5. Experiments and Results

Table-1 compares existing algorithm & Starvation-Removal algorithm used with Lease Workload-Format (LWF) files in which we have used first lease as BE with highest time requesting and other all leases are used as AR.

Table-1 uses P, X, Y & Z parameters where P= Sample size (Number of leases in LWF file), X= Start Date & Time of First BE lease in our experiment, Y= End Date & time of First BE lease in our experiment, Z= Number of times suspension of First BE lease in our experiment.

P	Algorithm used	X	Y	Z
10	Existing	25-06-2011 13:01:00	26-06-2011 13.01:00	2
	Proposed	25-06-2011 13:01:00	26-06-2011 00.31:32	2
20	Existing	25-06-2011 13:01:00	26-06-2011 10.23:40	5
	Proposed	25-06-2011 13:01:00	26-06-2011 00.09:22	2
30	Existing	25-06-2011 13:01:00	26-06-2011 20:19:54	7
	Proposed	25-06-2011 13:01:00	26-06-2011 19.34:34	2
40	Existing	25-06-2011 13:01:00	26-06-2011 13:30:44	11
	Proposed	25-06-2011 13:01:00	26-06-2011 10.51:16	2
50	Existing	25-06-2011 13:01:00	26-06-2011 09:27:00	16
	Proposed	25-06-2011 13:01:00	26-06-2011 5.38:00	2

Table 1. Comparison between Existing & Starvation-Removal Algorithm

Figure-3 shows graph that represents efficiency of Starvation-Removal algorithm proposed by us over existing algorithm. Table -2 shows sample size of our experiments with AR-to-BE Conversion algorithm, first column named P shows number of leases in LWF file. Second and third columns show number of lease rejected due to unavailability of computing resources in existing algorithm and our proposed algorithms respectively. Here for experimental purpose, we took 4 nodes available, first BE lease & then rest all AR leases. Figure-4 shows a graph which depicts efficiency of our algorithm AR-to-BE Conversion over existing algorithm.

In the experimented runs suspension time & resumption time is neglected, but that time can't be neglected in actual runs when overheads and bad bandwidth worse the running of BE leases.

6. Conclusions

To reduce request rejection rate between consumer and provider, and increase resource utilization on cloud provider side, Starvation-Removal and AR-to-BE Conversion algorithms are necessary. Proposed Starvation-Removal algorithm applies technique that provides a maximum limit a lease can be suspended considering constraints' flexibilities to maximize the chances of their acceptance. Using proposed algorithms, consumers will get suitable lease and their allowable suspension according to their needs. AR-to-BE Conversion algorithm will reduce consumers' efforts to wait for exact time of lease execution and checking weather lease is provisioned or rejected at all. These algorithms will not handle the situations when system has multiple requests of same type of lease for a single slot and therefore, it will just follow first in first out queue to handle them as proposed in Haizea.

Future work may help in situations where system needs to take care of more priority BE leases and insert BE leases in queue according to their priority. This work suggest that, a new type of BE lease can be introduced that can handle requests according to their priority factors, suspensions and dead line of time to finish.

```
Algorithm 1: STARVATION REMOVAL ALGORITHM
if scheduling lease type= = Immediate lease then:
    if resources required are available at the time then:
        allocate resources to Immediate lease
    else
        reject Immediate lease and print message
else
    if scheduling lease type= = BE then:
        Queue BE lease and set state of lease to queue.
    else
        if scheduling lease type= =AR then:
            if BE lease request already scheduled at that time then:
              if number of suspension > max limit provided by user in
              advance for that BE then:
                Do not schedule this AR & reschedule BE to be
                suspended on slot vacated by AR
              else
                Increase counter of suspension of that BE and
                allocate resources to AR lease
            else
              if resources are not available as being used by other AR
              lease request then:
                reject the AR lease and print message
        else
            if scheduling lease type= = DLS then:
                slack=(deadline-start time) / duration
                if slack < 1.1 then
                 reject lease and print message to extend dead line or
                 submit lease as AR lease
                else
                 find a single time slot which can satisfy complete
                 lease within deadline
                 if new lease is not schedulable as above then:
                     find multiple slots which together can satisfy
                     this conditions
                 if new lease is not schedulable by any of the above
                 methods then:
                   find leases to be rescheduled & reschedule
                   deadline leases
                 else
                   reject new lease & print message
            else
                print message invalid lease type
```

fig. 1. Algorithm for Starvation-Removal.

```
Algorithm 2: AR-TO-BE CONVERSION ALGORITHM
if scheduling lease type= = Immediate lease then:
    if resources required are available at the time then:
        allocate resources to Immediate lease
    else
        reject Immediate lease and print message
else
    if scheduling lease type= = BE then:
        Queue BE lease and set state of lease to queue.
    else
     if scheduling lease type= =AR then:
            if no AR lease is running on required time slot & resources are
            free then:
                allocate resources to AR
            else
                if AR not getting resources & going to be rejected by the
                scheduler then:
                  Provide dynamic choice for AR lease convertible to
                  BE lease
                  if accepted then:
                      convert this AR to BE, queue BE lease and set
                      state of lease to queue
                  else
                      reject AR.
     else
            if scheduling lease type= = DLS then:
                slack=(deadline-start time) / duration
                if slack < 1.1 then
                 reject lease and print message to extend dead line or
                 submit lease as AR lease
                else
                 find a single time slot which can satisfy complete lease
                 within deadline
                 if new lease is not schedulable as above then:
                     find multiple slots which together can satisfy this
                     conditions
                 if new lease is not schedulable by any of the above
                 methods then:
                   find leases to be rescheduled & reschedule deadline
                   leases
                 else
                   reject new lease & print message
            else
                print message invalid lease type
```

fig. 2. Algorithm for AR-to-BE Conversion.

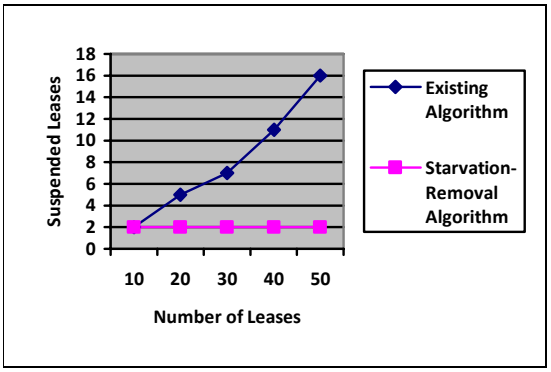

fig. 3 Graph Showing Efficiency of Starvation-Removal Algorithm over Existing Algorithm.

P	Existing algorithm No. of AR(s) rejected	Proposed algorithm No. of AR(s) rejected
10	6	0
20	8	0
30	12	0
40	16	0
50	20	0

Table 2. Comparison between Existing & AR-to-BE Conversion Algorithm.

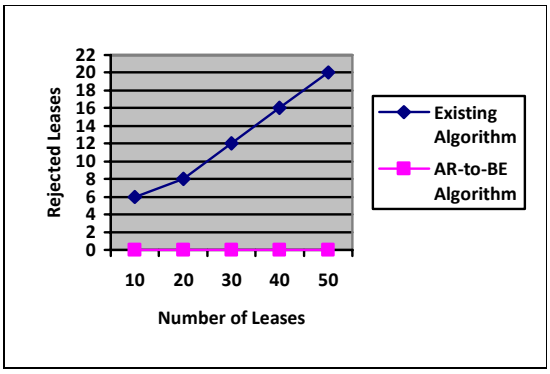

fig. 4. Graph Showing Efficiency of AR-to-BE Algorithm over Existing Algorithm.

References

[1]. M. Armbrust, A. Fox, R. Grith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, Above The Clouds: A Berkeley View of Cloud Computing, *Technical report UCB/EECS-2009-28, Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley,* USA, February 2009.

[2]. OpenNebula Pro, OpenNebulaPro White Paper, Rev20110126, https://support.opennebula.pro/attachments/token/coiuzlpxct7oyvq/?name=OpenNebulaPro_White_Paper_Rev20110126.pdf retrieved on 23-May, 2011.

[3]. OpenNebula Virtual Machine, http://opennebula.org/documentation:rel2.2:vm_guide retrieved on 22-May, 2011.

[4]. OpenNebula Scheduler, http://opennebula.org/documentation:archives:rel2.0:schg, retrieved on 22-May, 2011.

[5]. Haizea Lease Policies, retrieved from http://haizea.cs.uchicago.edu/whatis.html on 23-May, 2011.

[6]. J. Akhani, S. Chaudhary and G. Somani, Negotiation for Resource Allocation in IaaS Cloud, Proc. Fourth Annual ACM Bangalore Conference*, Banglore, India,* 2011.

[7]. A. Nathani, S. Chaudhary, and G. Somani, Policy based resource allocation in IaaS cloud, *Future Generation Computer Systems*, Jun. 2011.

[8]. M. Salehi and R. Buyya, Adapting Market-Oriented Scheduling Policies for Cloud Computing, *Algorithms and Architectures for Parallel Processing*, Springer, 2010, 351-362.

[9]. R. Buyya, D. Abramson, and S. Venugopal, The Grid Economy, *Proceedings of the IEEE, 93*(3), 2005*, 698–714*.

[10]. H. Mehta, P. Kanungo, and M. Chandwani, Performance Enhancement of Scheduling Algorithms in Clusters and Grids using Improved Dynamic Load Balancing Techniques, *Proc. 20th international conference companion on World wide web*, Hyderabad, India, 2011, 385-389.

[11]. D. Klus´a˘cek, L. Matyska, and H. Rudov´a, Local Search for Deadline Driven Grid Scheduling, Proc. *Third Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2007),* Czechia, 2007, 74–81.

[12]. B. Sotomayor, K. Keahey, I. Foster, and T. Freeman, Enabling Cost-Effective Resource Leases With Virtual Machines, *Hot Topics session in ACM/IEEE International Symposium on High Performance Distributed Computing 2007 (HPDC 2007)*, Monterey Bay California, 2007.

[13]. B. Sotomayor, R. Montero, I. Llorente, and I. Foster, Resource Leasing and the Art of Suspending Virtual Machines, *Proc. 11th IEEE International Conference on High Performance Computing and Communications (HPCC-09)*, Seoul Korea, June 2009, 59-68.

[14]. B. Sotomayor, R.S. Montero, I.M. Llorente, and I. Foster, Virtual infrastructure management in private and hybrid clouds, Internet Computing, IEEE 13(5), 2009, 14–22.

[15]. B. Sotomayor, R. S. Montero, I. M. Llorente, I. Foster, and F. de Informatica, Capacity Leasing in Cloud Systems Using the Opennebula Engine, *Cloud Computing and Applications 2008 (CCA08)*, volume 2008,

2008, 1–5.

[16]. B. Sotomayor, K. Keahey, and I. Foster, Combining Batch Execution and Leasing Using Virtual Machines, *Proc. 17th International Symposium on High Performance Distributed Computing (HPDC '08:),* ACM, Boston Massachussets, 2008, 87–96.

[17]. Z. Afoulki, J. Rouzaud-Cornabas LIFO, ENSI de Bourges, A Security-Aware Scheduler for Virtual Machines on IaaS Clouds, *Report 2011*, Universite D'orleans, 2011.

[18]. R. N. Calheiros, R. Ranjan, A. Beloglazov, D. Rose C.A.F., and R. Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, *Software: Practice and Experience,* 41 (1), 2011, 23-50.

[19]. A.S. Tanenbaum, Modern Operating Systems (3 International ed: Pearson Education, 2008).

Authors Biography

Vivek Shrivastava 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.