ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Efficient Scheduling Strategy for Dynamic Resource Allocation on IaaS Private Cloud

Shakkeera L and Latha Tamilselvan

Department of Information Technology, School of Computer, Information and Mathematical Sciences, B.S.Abdur Rahman University, Chennai- 600 048, India.

ARTICLE INFO

Article history:
Received 23 June 2015
Accepted 25 August 2015
Available online 2 September 2015

Keywords:
Cloud Computing, Resource Allocation, Task Scheduling, Haizea, OpenNebula, Resource Starvation Avoidance (RSA), Resource Provisioning Conversion (RPC), Infrastructure as a Service (IaaS).

ABSTRACT

Cloud computing is a new business model and service model. Its core concept is that it doesn't rely on the local computer to do computing, but a computing resources operated by third parties that provide computing, storage and networking resources. Infrastructure As A Service (IaaS) cloud service providers handle with a large set of computing resources. Scheduling in resource allocation mechanism for cloud computing is receiving more attention with increases of cloud popularity. Private cloud resources are provided to cloud consumers on demand manner in the form of virtual machines. The available resources are utilized proficiently without affecting the service parameters of cloud. The existing strategies for scheduling the data intensive applications can solve the problem of resource provisioning but none of the algorithms could be considered for different resources for multiple heterogeneous user requests. So, for efficient resource scheduling system, we need to deploy a resource provisioning mechanism in IaaS private cloud system that uses the resource scheduling algorithms in order to process the multiple tasks. The main aim of this work is to develop an efficient scheduling strategy for dynamic resource allocation on private cloud setup for multiple client requests. Evaluate the system based on the performance metrics like resource utilization rate, makespan, system throughput, time overhead, lease request acceptance and rejection rate. Our results show that it maximizes resource utilization rate, system throughput, and acceptance rate of leases and minimizes makespan, task completion time when multiple heterogeneous users requests are submitted to private cloud environment.

© 2015 AENSI Publisher All rights reserved.

To Cite This Article: Shakkeera L and Latha Tamilselvan, Efficient Scheduling Strategy for Dynamic Resource Allocation on IaaS Private Cloud. ***Aust. J. Basic & Appl. Sci.,*** *9(27): 168-180, 2015*

INTRODUCTION

In cloud computing (Armbrust *et al*., 2009; Tharam Dillion *et al*., 2010) a cloud is a group of distributed computers providing on-demand computational resources or services to the remote clients over a network. Cloud providers offer their services according to three primary service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). In an Infrastructure-as-a-Service (IaaS) (Aobing Sun *et al*., 2011) cloud, resources or services are provided to users in the form of leases (virtual machines). The users can access the cloud resources using virtualization solutions, e.g the KVM or Xen hypervisor. One of the advantages of IaaS clouds is that the computational resource capacities are provided to end-users in flexible and efficient manner.

In cloud computing, resource allocation (Zhen Xiao *et al*., 2013; Arfeen *et al*., 2011; Lin, 2013) is the mechanism of assigning available resources to the requested cloud applications over the internet to service consumers. Cloud resources can be seen as any resource (physical or virtual) that users may request from the cloud services. User can specify computational requirements such as CPU, memory and storage and network requirements such as bandwidth and delay. Due to the heterogeneous and time-variant features in a cloud, the resource provisioning mechanism becomes a complex task, forcing the dynamic resource allocation system to respond with minimal turnaround time in order to maintain the provider's service quality requirements. So, resource allocation starves the services if the allocation is not managed precisely. In section 3, defines the challenges or issues in resource allocation. Our proposed resource provisioning mechanisms solves the above problem by allowing the service providers to allocate the virtual machines upon physical resources for end users. The virtual machines are allocated according to the requirements specified by the end users. Mainly private cloud model is designed to demand to an organization that wants more control over their resources than they can get by using a third-party hosted service such as Amazon's Elastic Compute Cloud (EC2) and Simple Storage Service (S3).Unlike public clouds, which

Corresponding Author: Shakkeera L, Department of Information Technology, School of Computer, Information and Mathematical Sciences, B.S. Abdur Rahman University, Chennai- 600 048, India

deliver services to multiple organizations, a private cloud is devoted to a single organization.

Task scheduling is the method in which the tasks/processes are given access to system resources based on processor time and communication bandwidth. This method is usually done to balance the load of the systems effectively and finally achieve a Quality of Service (QoS). The goal of load balance is to create a balanced average resource utilization of the physical machines. Scheduling in the cloud system is an NP-complete problem. Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) (Guo *et al*., 2012), Min-Min, Max-, Min (Jemina and Arockiam, 2014) meta-heuristics methods are developed to schedule the tasks in different service- oriented environments. However, these heuristic optimization algorithms usually make some hypothesis and reduce the complexity of the problems to make computationally realistic.

Resource scheduling (Elghoneimy *et al*., 2012) is a process whereby resource providers allocate resources to users. For a cloud data center, the types of resource scheduling strategies must consider physical servers, physical server clusters, shared memories, bandwidths and virtual machine images. Dynamic resource scheduling performs three resource-related operations: (1) it calculates the demand of resources that each VM should request based on the reservation and shares settings and constraints for VMs, as well as resource pool nodes; (2) it does initial placement of VMs on to hosts automatically; and (3) it suggests and performs live VM migration to do load balancing across hosts in a dynamic environment when the VMs resource demands vary during a period of time. Resource management (Sunilkumar and Gopal, 2014) is a broader concept than resource scheduling, because in addition to resource scheduling, it also includes resource monitoring, resource provisioning, resource mapping, resource adaptation, automatic installation and configuration and the implementation of scheduling tasks. The section 4 describes proposed dynamic resource allocation system on IaaS cloud with resource allocation policies and resource provisioning mechanisms.

Virtualization (Sotomayor *et al*., 2009) means that the computing components run in a virtual environment rather than in a physical server. Virtual machine scheduling (Kruekaew and Kimpan, 2014) is the highest strategy in scheduling management, and must be chosen by the data center owner and managers. The main aim is to fulfill the goals of the scheduled resources and to provide strategies when resources cannot satisfy all real-time requirements. In our approach virtual machines are considered as leases for resource allocation.

Open Nebula (Borja Sotomayor *et al*., 2009) is an open source cloud service framework (IaaS) to establish private cloud middleware environment. It allows the user to deploy and manage virtual machines on physical resources. It can set user's data centers or clusters as flexible virtual infrastructure that can automatically adapt to the change of the service load. Using Opennebula we can establish public, private and hybrid cloud. OpenNebula can also work with Haizea resource scheduler tool. OpenNebula (Sotomayor *et al*., 2009) uses only immediate lease provisioning to schedule IaaS cloud resources using match-making algorithm. The match-making algorithm allocates resources with a higher rank expression to allocate virtual machines on physical nodes.

Haizea is an open source resource lease manager that tries to address the issues with complex resource allocation policies. Haizea uses resource leases (Sotomayor *et al*., 2009) as resource allocation abstraction and implements these leases by allocating virtual machines (VMs). We define a lease as a negotiated and renegotiable agreement between a resource provider and a resource consumer (end users), where the former agrees to make a set of resources available to the latter, based on a set of lease terms presented by the resource consumers. When user wants to request computational resources, the leases are then implemented as VMs. Haizea supports four kinds of resource allocation policies: advanced reservation, best effort, immediate and deadline time leases. The lease terms must encompass the following:

- Hardware resources consumed by the clients, such as CPUs, memory, and network bandwidth.
- Software application services required on the leased resources.
- Availability period during which a user requests that the hardware and software resources be available.

RELATED WORK

Many scheduling strategies are used for dynamic resource allocation in cloud computing scenarios. It uses different approaches and algorithms which are discussed below: Each one of them have their own policies to efficiently utilize the data center resources.

The work (JieHui *et al*., 2014) combines optimization algorithms like Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) for task scheduling. Initially, PSO schedule the applications to cloud resources by considering computation and data transmission cost. The ant colony optimization algorithm finally gets the optimal results for task scheduling. This algorithm performs to get optimized solutions in real-time scenarios. These algorithms minimize the execution cost or applications completion time.

Scheduling algorithms mainly aims to reduce the application execution time/cost in resource constraint cloud environment. Meanwhile, the datacenter resources must be efficiently utilized by the all the

application user requests. This paper (Gang Zhao, 2014) describes cost- aware fitness function by considering the cost of the resources usages in PSO algorithms. This approach does consider the idle resources for multiple requests distribution on scheduling.

The research work (Jiayin *et al*., 2011) describes the adaptive resources allocation algorithm with preemtable tasks. Static task scheduling like Adaptive List Scheduling (ALS) and Adaptive Min-Min Scheduling (AMMS) algorithms are used for only static resource allocation. It executes the applications by considering all the short and long tasks. In this method, scheduler is re- calculating the remaining static resources with predefined frequency with ending time of all the submitted tasks.

The paper (Amit *et al*., 2012) discusses policy based resource allocation on Infrastructure as a Service (IaaS) cloud using Haizea scheduler. Many of the IaaS cloud resource management use simple immediate and best- effort policies. Based on this policy, many of the services requests are rejected or placed into queue due to insufficient VM resources in data center service providers. The proposed method is implemented in Haizea that accepts new lease requests and process the requests based on the resource availability on cloud.

The paper (Bhilare and Vivek, 2012) describes starvation- removal algorithm for resource scheduling. This algorithm fixes the max-limit value for suspending the best effort lease requests. Using this approach, the priority based user requests (advanced reservation) are processed with limited suspensions of best effort leases. Advance reservation-to-Best effort conversion algorithm reduces consumers' efforts to wait for exact time of lease execution. These algorithms will not handle the situations when system has multiple requests of same type of lease for a single slot and it will just follow first in first out queue to handle them as proposed in Haizea. And AR-to-BE conversion algorithm will lead to starvation problem again and again.

Leases for resources are static in nature. Once resources are allocated to a lease request, these resources cannot be altered during the complete lifetime of the lease request. This contradicts with the philosophy and implementation of a pure on-demand elastic cloud where resources of a lease are continuously monitored against the utilization and altered based on the requirements. The paper work (Chokhani and Somani, 2013) aims at the inclusion of three important features to mitigate the above said shortcomings in Haizea. First, it introduces a new class of lease: Dynamic lease to accommodate resource changes. Second, it examines virtual machine resource utilization to decide about the demand and need of allocation change and third, it accommodates the expected changes in resource allocation by introducing two new sub-leases which allow dynamic resource allocation. The above method introduces the feature of on-demand resource availability as well as elasticity using lease auto-negotiation in Haizea.

PROBLEM STATEMENT

Application processing in cloud data centers creates the challenge of an on-demand resource provisioning and allocation for dynamic workload requests. Some of the research challenges or issues related to resource allocation are mentioned in the following statements: Most of the existing algorithms focus on optimization of physical resources to virtual resources and migrate the applications to another resource to manage load balancing and increases the resource utilization. But in these approaches, the suspension of applications is occurred due to the shutdown of the virtual machines. When the workload shoots up, then the existing virtual machines are shutdown and new virtual machines with increased capacity of physical resources are assigned. This leads to suspension of current running client applications. Also, at minimal load, many of the virtual machines are unused leading to poor resource utilization and unnecessary user usage cost. Based on resource allocation policies, cloud service providers allocate resources to competing requests. Sometimes it is not possible for cloud service providers to satisfy all the lease requests which come to them on immediate basis due to lack of resources in data centers. By using static resource allocation scheme to assign fixed resources to lease request, then the application may be slowed down due to insufficient resource utilization. So, our proposed dynamic resource allocation system on private cloud focuses to maximize resource utilization, minimizes make span and increases the acceptance of more number of lease policies when the workload changes dynamically.

DYNAMIC RESOURCE ALLOCATION AYSTEM (D-RAS) FOR APPLICATION PROCESSING ON IAAS PRIVATE CLOUD

Cloud computing is on demand strategy as it offers flexible resource allocation for reliable and guaranteed services in pay as-you-use manner to users. Cloud computing involves multiple cloud users requesting number of cloud services simultaneously. To satisfy their need, there must be a provision that all resources are made available to requesting user in efficient manner. Based on resource allocation policies, cloud resource provider allocates resources to competing requests.

System Model:

In this section we mention the system model used throughout this research work. In our system model, resources are provisioned completely from private clouds. The proposed resource allocation system is developed using Open Nebula open source software to setup and run the private cloud. A cloud

controller component manages the pool of resources via a provisioning resource allocation policy, that is, a policy that decides when to lease and to release resources from IaaS clouds. Parallel or sequential task requests are processed in our proposed approach. We assume that end users send their multiple heterogeneous task requests to a Haizea scheduler, which enqueues tasks and assign them to the pool of available resources based on resource allocation strategies. And also assume that a provisioning task request issued to the private cloud will sustain delays that depend on the process used by the IaaS to select, lease-and-run, and release (shutdown) a VM instance resource. To request resources from cloud, the user has to submit their lease requests in a specific format to haizea scheduler. Main parameters in this lease request are type of resource lease policy, number of virtual or physical nodes, start time, duration for processing tasks and deadline time. CPU and Memory are considered as computational resources, which can be requested by the users.

Dynamic Resource Allocation System (D-RAS):

A Dynamic-Resource Allocation System (Shakkeera and Latha , 2013) on IaaS cloud is an important strategy that aims to guarantee that the application requirements are met correctly by the service provider's infrastructure. Along with this guarantee to the user, resource allocation mechanisms are also consider the current status of each resource in the cloud environment, in order to apply resource provisioning mechanisms to better allocate physical and/or virtual resources to user's applications, thus minimizing the operational cost of the cloud environment. It is important to note that the clients and developers may see those finite resources as unlimited and the tool that will make this possible is the D-RAS. The D-RAS deals with these volatile requests in an elastic and transparent way. This elasticity allows the dynamic use of physical resources, thus avoiding both the under-provisioning and over-provisioning of resources. When applying parallel processing in executing the tasks, always need to consider how to allocate resources to tasks and how to schedule overheads when VMs prepare, terminate or switch the tasks.

Resource allocation strategy can solve the above mentioned problems. The framework model of our proposed dynamic resource allocation system for multiple workflows on IaaS private cloud is shown in Fig. 1. This proposed system aims to build the private cloud environment using OpenNebula (execution engine) with Haizea resource scheduler or maneger. This model is divided into three different phases namely.

1. Resource discovery
2. Resource allocation policies
3. Resource provisioning mechanism

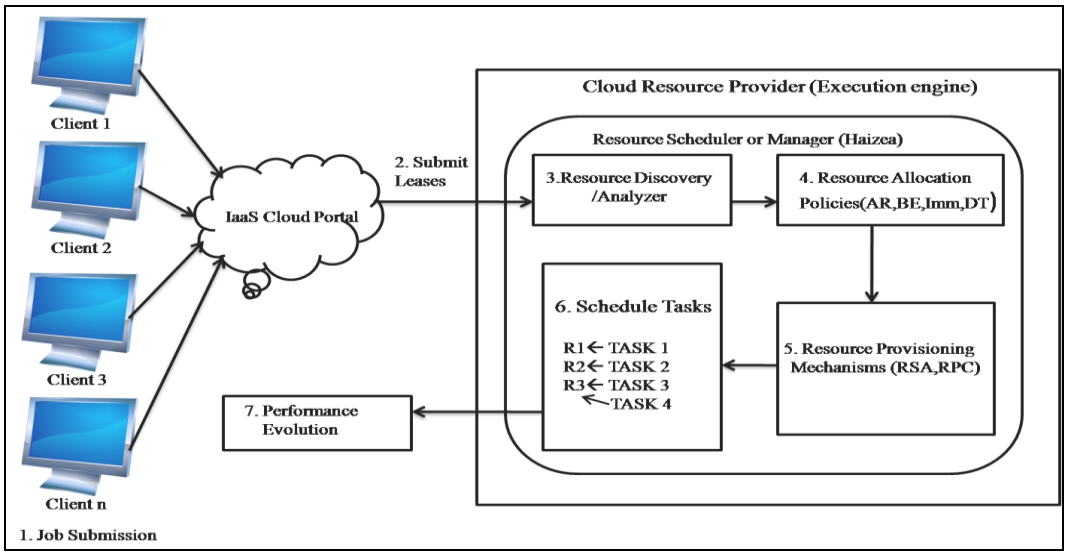

Fig. 1: Dynamic Resource Allocation System (D-RAS) on IaaS Private Cloud.

Resource Discovery:

In resource discovery, cloud controller determines the VM resources present in the network systems and collects the status information related to them. OpenNebula dashboard shows the details of the resources present in the cloud like number of hosts, number of virtual machines, CPU usage, memory usage, network transmission rate, network reception rate and resource current status.

Resource Allocation Policies:

The resource allocation supports four different policies of renting the computing capacity from a cloud service provider as leases in Haizea. Lease manager simultaneously check the resource limitations in cloud data center. Formats for

specifying lease requests in Haizea can be shown as below:

Advance Reservation (AR): The VM resources are reserved in advance. Resources should be available at a specific time.

Best-Effort (BE): Resources are provisioned and client requests are placed in a queue when resources are not available.

Immediate: When a client submits a request, either the resources are provisioned immediately, or the request is rejected, based on the resource availabilities.

Deadline Time (DT): This type of lease requests must be granted before a deadline time period or otherwise lease requests are terminated.

Table I: Shows resource allocation policy report for different types of resource leases.

The following XML based files describes Sample Dynamic Lease Request for AR leases.

Sample- Dynamic AR Lease Request

```
<lease-workload name="AR-sample-lease-request">
<description>
</description>
<lease-requests>
   <!-- First lease request-->
     <lease-request arrival="00:00:00">
       <lease preemptible="false">
       <nodes>
          <node-set numnodes="1">
             <res type="CPU" amount="100"/>
             <res type="Memory" amount="1024"/>
         </node-set>
       </nodes>
       <start time="00:30:00"/>
       <duration time="01:00:00"/>
       <software>
          <disk-image id="foobar.img"size="1024"/>
       </software>
      </lease>
    </lease-request>
</lease-requests>
```

VM scheduler/lease manager reserves the resources for AR lease. Whenever start time of a reservation comes, scheduler allocates resources to users in the form of VMs, which are deployed on physical machines. Lease manager assumes that BE leases are preemptable and they do not have any time constraints. Immediate and advance reservation leases are non-preemptable and have time constraints, such as start time and end time. It preempts BE leases whenever the resources are required for AR or immediate leases. If the systems have more number of advance reservation and immediate leases then best effort leases are not have enough resources for completion within a certain time limit (starvation and resource rejection (Heba *et al*., 2014)). Deadlines can be associated with best-effort leases for those consumers who want to get resources (for local users) for their best-effort leases within certain time limit (Mohsen *et al*., 2014). These kinds of leases can be considered as deadline time leases. Deadline time leases are assumed to be preemptable but there is a limitation to their preemptability.

Resource Provisioning Mechanisms:

Resource scheduling in real time cloud systems is a challenging task, as it involves coordinating multiple computational sites for resource sharing and scheduling in an efficient manner. The objective of the resource provisioning mechanism is to increase the availability of resources and maximizes the efficient resource utilization rate. When managing a private cloud with limited resources, an immediate provisioning model is insufficient. A lease-based resource provisioning model that can act as a scheduling back-end for OpenNebula, supports other provisioning models other than the immediate lease models in existing cloud providers. In particular, Haizea adds support for both best-effort provisioning and advance reservations, when managing a finite number of resources.

To evaluate and analysis the performance of the dynamic resource allocation system, the following task cases are considered which are related to QoS based heuristic algorithms for scheduling.

Case 1: Tasks are randomly chosen.

Case 2: A few long tasks along with many short tasks.

Case 3: A few short tasks along with many long

tasks.

Haizea scheduler is based on lease-based resource provisioning concept. In that AR and Immediate leases preempt the BE leases because AR and Immediate leases have higher priority that BE lease. The following algorithm 1 (Lease-Based Resource Provisioning Strategy) works on the above mechanism.

Table I: Resource Allocation Policy Report.

Resource Lease Policy	Arrival Time (hh:mm:ss)	Preemptiable	No.of Nodes	CPU Speed	Memory	Start Time (hh:mm:ss)	Duration Time (hh:mm:ss)	Deadline Time (hh:mm:ss)
AR	00:00:15	False	1	100	1024	00:30:00	01:00:00	-
BE	00:00:00	True	4	100	1024	-	00:30:00	-
Imm	00:00:15	True	1	50	1024	Now	00:15:00	-
DT	00:20:15	True	1	100	1024	00:40:00	00:10:00	02:00:00

Algorithm 1: Lease-Based Resource Provisioning Strategy

```
1   IF request lease type== Immediate THEN:
2       IF resources required are available at the requested time THEN:
3           Allocate the resources to Immediate lease
4       ELSE
5           Reject Immediate lease
6   ELSEIF scheduling lease type== BE THEN:
7       IF resources required are not available at that time THEN:
8           Queue BE leases and set state of lease request to queue
9   ELSEIF scheduling lease type==AR THEN:
10      IF resources required are available at the requested time THEN:
11          Allocate the resources to AR
12  ELSEIF BE is scheduled at the required time THEN:
13      Suspend BE lease
14      Allocate the resources to AR lease
15  ELSEIF Immediate leases are scheduled at the required time THEN:
16      Reject the AR lease request
```

The following section describes the algorithms that can be used to solve the problems in resource provisioning mechanisms.

Problems in Resource Provisioning Mechanism:

Problem 1: 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 request is based on 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. Algorithm 2 describes the steps involved in the Resource Starvation Avoidance (RSA). This algorithm successfully overcomes the above problem and increases chances of accepting BE lease request to finish in a small time interval by limiting number of suspensions.

Algorithm 2: Resource Starvation Avoidance (RSA)

Input: No.of Lease types like Advanced Reservation(AR), Best-Effort(BE), Immediate and Deadline Time (DT) Leases

```
1       BEGIN
2         IF request lease type==Immediate THEN:
3           IF resources required are available at that time
            THEN:
4             Allocate resources to Immediate lease
5           ELSE
6             Reject Immediate lease
7         ELSEIF request lease type==BE THEN:
8           IF resources required are not available at that
            time THEN:
9             Queue BE leases and set state of lease
              request to queue
10        ELSEIF scheduling lease type==AR THEN:
11          IF old BE lease request already scheduled at
```

```
        that time THEN:
12          IF number of suspension>max-limit for
            that BE THEN:
13              Do not schedule this AR & continue
                with BE lease
14      ELSE
15          Increase counter of rejection of that BE
            and allocate resource to AR lease
16      ELSEIF another AR lease request already
        scheduled at that time THEN:
17          Provide dynamic choice for new AR lease
            resource provisioning
            conversion into BE lease
                    //Apply RPC algorithm
18      IF BE leases are accepted THEN:
19          Convert this new AR to BE, queue BE
            lease request and set state of lease to
            queue
20      ELSE
21          Reject new AR
22      ELSEIF request lease type=DT THEN:
23          slack=(deadline-start time)/duration
24      IF slack<1.0 THEN:
25          Reject lease and extend deadline or submit
            lease as AR lease
26      ELSE
27          Find a single time slot which can satisfy
            complete lease within deadline
28      IF new lease is not schedulable as above
        THEN:
29          Find a multiple slots which together can
            satisfy this conditions
30      IF new lease is not schedulable by any of
        the above methods THEN:
31          Find leases to be rescheduled &
            reschedule deadline leases
32      ELSE
33          Reject new DT lease
34      ELSE
35          Invalid lease type
36      END
```

The objective of Resource Starvation Avoidance algorithm is to prevent BE leases from starvation of resources by denying new AR leases. The algorithm will not schedule the AR leases after a fixed number of suspension of BE leases. The algorithm maximizes resource utilization rate, BE lease request acceptance rate, system throughput and minimizes the time overhead. The final results are measured using the following performance metrics equations (1) (2) (3) (4) (5). The max-limit value in proposed algorithm is calculated using the equation (6).

Resource Utilization Rate (RUR) – Resource utilization rate is ratio of total utilization of resources with total capacity of resources in system. Average utilization of all physical/virtual servers (CPU, memory utilization and network bandwidth) must be same to achieve system scalability.

$$RUR = \frac{\text{Total utilization of resources}}{\text{Total capcacity of resources in system}} * 100 \quad (1)$$

Total utilization of resources is based on computation time of lease requests.

$$\text{Computation Time} = \sum_{i=1}^{n} m(LR_i) . t(LR_i) \quad (2)$$

Where 'n' is the number of leases, $m(LR_i)$ is the number of VM's in lease request LR_i , $t(LR_i)$ is the time of lease LR_i.

BE Lease Request Acceptance Rate (BE Lease RAR) – BE lease request acceptance rate is given by

$$\text{BE Lease RAR} = \frac{\text{Number of times First BE lease suspended}}{\text{Total number of leases}} * RUR \quad (3)$$

Where RUR is Resource Utilization Rate.

Australian Journal of Basic and Applied Sciences, 9(7) August 2015, Pages: 110-112

Task Completion Time – Task completion time is calculated amount of time required for any particular task to be completed. Task completion time is calculated by the following equation (4)

$$\text{Task Completion Time} = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} \sum_{j=1}^{m} p_{i,j} \quad (4)$$

System Throughput- System throughput is the total number of tasks that complete their execution per time unit in cloud resources. The throughput of a system is the number of jobs or tasks, they system completes per unit time. The improved system throughput must be obtained using dynamic resource provisioning mechanisms.

Time Overhead – The time overhead for VM lease request completion is estimated based on lease request suspension time, lease request resumption time and time spent in queue. Time overhead for each lease request is calculated by the following equation (5)

$$\text{Time Overhead} = \sum_{i=1}^{n} \text{LRST} + \sum_{i=1}^{n} \text{LRRT} + \text{Time spent in queue} \quad (5)$$

Where

$\sum_{i=1}^{n} \text{LRST} - \text{Lease Request Suspension Time}$

$\sum_{i=1}^{n} \text{LRRT} - \text{Lease Request Resumption Time}$

Here, n is the total number of suspension and resumption of that VM lease request.

Lease request time overhead of suspension and resumption times are calculated by using T_S and $T_{R..}$

The overhead of suspension time is

$$T_S = \frac{\text{VM} - \text{Mem}}{S}$$

The overhead of resumption time is

$$T_R = \frac{\text{VM} - \text{Mem}}{R}$$

Where, VM-Mem is the amount of VM memory, S is the rate of suspending megabytes of VM memory per second and R is the rate of resuming megabytes of VM memory per second.

Max-Limit – A max-limit value is based on the ratio of total number of current pending BE leases and running AR leases with system resource limitations.

$$\text{Max} - \text{Limit} = \frac{\text{No.of pending BE leases} + \text{No.of running AR leases}}{\text{Resource limitations}} \quad (6)$$

Problem 2: Problem occurs 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. Resource Provisioning Conversion (RPC) algorithm presented in this paper successfully overcomes this situation and thus provides proper usage of resources and increase the efficiency of resource provisioning by reducing request rejection rate.

Algorithm 3: Resource Provisioning Conversion (RPC)

Input: No.of Lease types like Advanced Reservation(AR), Best-Effort(BE), Immediate and Deadline Time (DT) Leases

```
1   BEGIN
2     IF request lease type==Immediate THEN:
3       IF resources required are available at that time
        THEN:
4         Allocate resources to Immediate lease
5       ELSE
6         Reject Immediate lease
7   ELSEIF request lease type==BE THEN:
8       IF resources required are not available at that time
        THEN:
9         Queue BE leases and set state of lease request
          to queue
10      ELSEIF scheduling lease type== new AR THEN:
11        IF no old AR lease request is running on required
          time slot & resources are free THEN:
12          Allocate resources to new AR lease request
13        ELSEIF old AR lease is running on required time
          slot and new AR lease not getting enough
          resources & going to be rejected by the scheduler
          THEN:
14          Provide dynamic choice for new AR lease
            resource provisioning conversion into BE
```

```
                lease
15      IF BE leases are accepted THEN:
                    //based on max-limit value
                    // Apply RSA algorithm
16          Convert this new AR to new BE, queue
            BE lease request and set state of lease to
            Queue
17          IF old BE lease request already
            scheduled at that time THEN:
18              Do not schedule this new BE &
                continue with old BE lease
19          ELSE
20              Increase counter of rejection of that
                old BE lease and allocate resources
                to new BE lease
21      ELSE
22          Reject new AR
23      ELSEIF request lease type=DT THEN:
24          slack=(deadline-start time)/duration
25  IF slack<1.0 THEN:
26      Reject lease and extend deadline or submit
        lease as AR lease
27  ELSE
28      Find a single time slot which can satisfy
        complete lease within deadline
29  IF new lease is not schedulable as above
    THEN:
30      Find a multiple slots which together can
        satisfy this conditions
31  IF new lease is not schedulable by any of the
    above methods THEN:
32      Find leases to be rescheduled & reschedule
        deadline leases
33  ELSE
34      Reject new DT lease
35  ELSE
36      Invalid lease type
37  END
```

The objective of the above algorithm is to prevent AR lease rejection by converting AR lease to a BE lease, and thus queues lease to be rejected to run in future time frames. If already old BE leases are scheduled then again dynamic max-limit value is calculated to reduce the number of suspension of queued BE leases. The RPC algorithm minimizes makespan and AR lease request rejection rate and maximizes system utilization. The final results are measured using the following performance metrics equation (7) (8) & (9).

Makespan **-** The total amount of time required to complete a group of jobs or total completion time taken to allocate all the tasks to a resource. The minimum makespan is obtained by using equation (7).

$$M = \sum_{j=1}^{N} \frac{pj}{m} \quad (7)$$

Here, N is the number jobs submitted in to the system, P_j is the processing time for the jobs and m is the number of the machines.

System Utilization, Accepted and Rejected Requests:

a. System Utilization

Here system utilization (U) is calculated as

$$U = \frac{\text{Used}_{\text{resources}}}{\text{Total}_{\text{resources}}} \quad (8)$$

Here Used_resources and Total_resources are calculated as follows,

If n leases are submitted to the system then

$$\text{Total}_{\text{resources}} = \text{Total}_{\text{CPU}} * (\text{lease}_{n}.\text{deadline} - \text{lease}_{1}.\text{start})$$

$$\text{Used resources for each lease} = \text{lease.CPU} * \text{lease.duration})$$

$$\text{Used}_{\text{resources}} = \text{Used}_{\text{resources}} + \text{Used resources for each lease}$$

b. Number of requests accepted

c. Number of requests rejected

Australian Journal of Basic and Applied Sciences, 9(7) August 2015, Pages: 110-112

AR Lease Request Rejection Rate (RRR) – AR lease request rejection rate is given by,

$$\text{AR Lease Request Rejection Rate(RRR)} = \frac{\text{Number of AR Leases rejected}}{\text{Total number of submitted Leases}} * \text{RUR} \quad (9)$$

RESULTS AND DISCUSSIONS

Experimental Setup:

The Ubuntu 12.04.1 64 bit server edition operating system (with Intel core 2 duo 2.2GHz processor speed, 3 GB RAM, 320GB hard disk) is installed in all machines to support the virtualization of the resources using the hypervisor like KVM which provides the KVM virtualization for creating the virtual machines. OpenNebula 3.6.0 open source cloud controller is installed in ubuntu operating system (front end machine). VT enabled ubuntu machines are acting like node controllers (with Intelcore i3 2.40GHz processor, 4GB RAM, 500GB hard disk). In our experiments, Haizea resource lease manager act as a VM scheduler for OpenNebula by modifying its VM scheduler component. In Haizea, input of leases (request for resources) can be done with the help of XML based files and this XML files can be given as input to Haizea lease manager. We have performed all the experiments on IaaS clouds with middleware and different virtualization resources. The experiments are carried out by generating the task requests from 1 to 500 (batch processing). Similarly, we have considered the cloud resources (physical or virtual machines) from 1 to 1000. In application processing, the tasks are chosen based on different task cases.

Results and Discussions:

The proposed approach compares the resource scheduling results with existing resource scheduling algorithms such as Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) (Lin, 2013; Mohsen Amini Salehi, 2014), Min-Min and Max-Min (Elghoneimy, 2012) in IaaS cloud based scenarios. Furthermore, the proposed algorithm compares QoS performance metrics like resource utilization rate, makespan, system throughput, time overhead and overall system utilization based on accepted and rejected leases requests. Table II. contains commonly used computing nodes specifications in cloud servers.

Table II: Computing nodes specifications

Lease Request ID	Processors	Memory	Hard disk Capacity
0	1	256	10
1	1	512	15
2	1	1024	20
3	1	2048	20
4	2	1024	20
5	2	2048	40
6	4	2048	80
7	4	4098	160

Resource Utilization Rate (RUR):

The collections of input lease requests (user's resource requests) are given to the resource scheduler. From the Fig 2, it is seen that the proposed RSA approach gives the better utilization of resources using resource provisioning mechanism in all the three different task cases. As lease requests varies from 1 to 500 there is increase of utilization rate in RSA algorithm compare to other heuristics algorithms. The resource utilization rate is maximized in all the three different task cases because lease request rejection rate is less and RUR is based on the collection of input leases and max-limit suspension value for incoming requests.

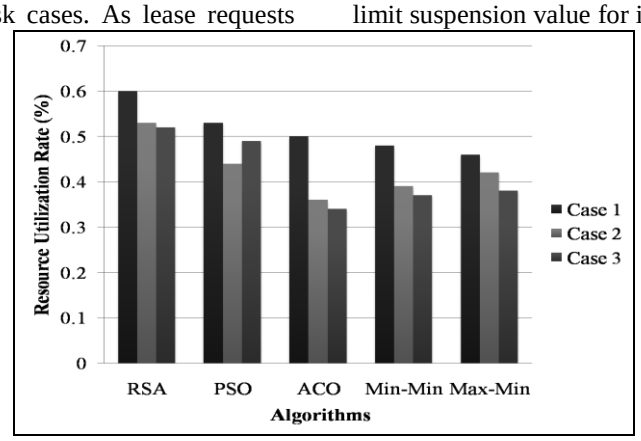

Fig. 2: Average Resource Utilization Rate.

Makespan:

From the below Fig 3, it is seen that the RPC technique gives minimum makespan value in all three different cases. In all the cases, RPC algorithm minimizes the makespan value compare to other algorithms. It shows that the proposed algorithm increases the chance of accepting more number of lease requests to finish in a small time interval by limiting number of suspensions and rejections of incoming requests.

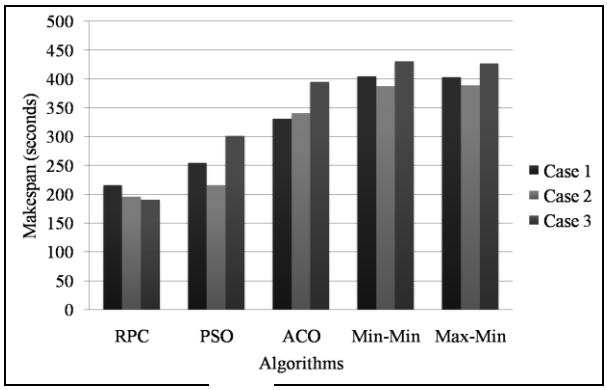

Fig. 3: Makespan

System Utilization:

System utilization rate values are observed and given in Fig 4. Utilization of resources based on lease start time, deadline time and duration time. The submitted input lease requests vary from 1 to 500, the graph shows that, the proposed algorithm efficiently uses the resources and it maximizes the system utilization, BE lease request acceptance rate and minimizes AR lease request rejection rate. BE lease request acceptance rate and AR lease request rejection rate mainly based on resource utilization rate in private cloud system. The proposed system accepts more number of lease requests dynamically.

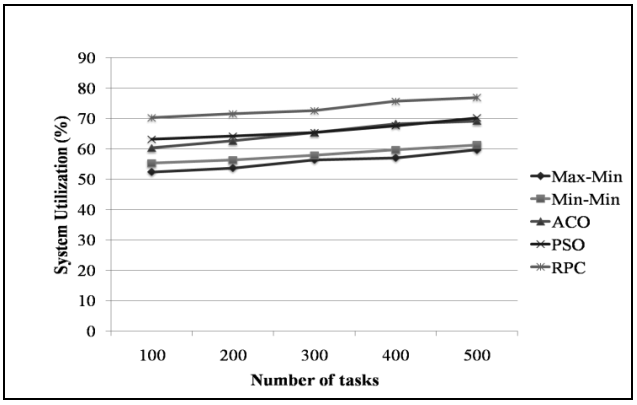

Fig. 4: System Utilization.

System Throughput:

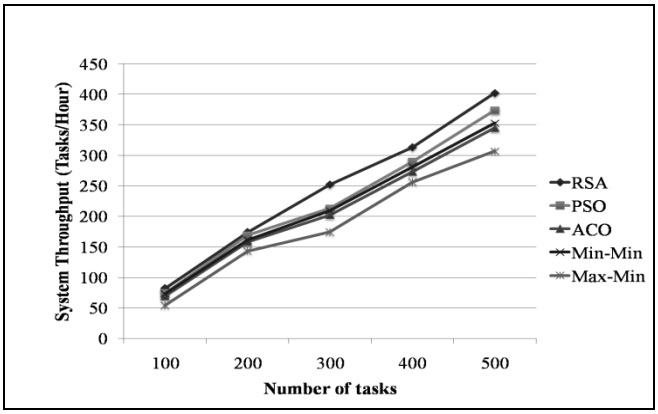

Fig. 5: System Throughput.

System throughput for application processing on IaaS cloud is observed in the Fig. 5 in the form of how many number of tasks are completed per hour duration.

In above case we have considered 1 to 500 tasks to evaluate the system throughput effectively.

The graph shows that, task scheduling using RSA algorithm improves system throughput because resource utilization and overall system utilization rates are high in resource constraint cloud servers. When number of tasks increases, the throughput of the system gradually decreases.

To overcome this problem, dynamically more number of physical/virtual resources are managed to balance the load in the system.

Time Overhead:

Submissions of lease requests are based on computing node specifications. From the below Fig 6, is observed that RSA approach having minimum time overhead and it achieves reduced average latency for all task's lease request. By this proposed approach for collection of tasks, time overhead is minimized compare to other algorithms. This is because the algorithm accepts number of lease requests in the form VMs and increases chances of lease requests acceptance rate by limiting number of suspensions.

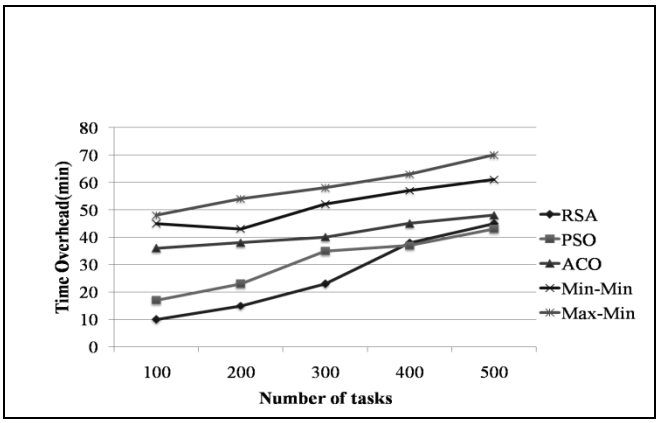

Fig. 6: Time Overhead.

Conclusion:

Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software and information provided to users over the network. It is an on demand service because it offers dynamic flexible resource allocation for reliable and guaranteed services in pay as you use manner to users. This research work discusses about efficient dynamic resource allocation system in private cloud scheduling environment. From all the experiments we have concluded that resource scheduler is most important in cloud environment. The existing algorithms do not consider the dynamic max-limit value for suspension of leases and the queued BE leases are not scheduled properly. The existing algorithms do not handle the situations when system has multiple heterogeneous end user requests. To overcome the above problems mentioned, we have designed the efficient dynamic resource allocation system that uses resource provisioning mechanisms effectively. When multiple leases requests are submitted the system handles the execution of the lease requests gradually. So our proposed approach improves the overall performance of dynamic resource allocation system in private cloud model. In future work, ideal resources should be considered to schedule and balance multiple client requests in energy-aware mobile cloud environment.

REFERENCES

Amit Nathani, Sanjay Chaudhary and Gaurav Somani, 2012. Policy based resource allocation in IaaS cloud. Future Generation Computer Systems, 28(1): 94-103. DOI:10.1016/j.future.2011.05.016.

Aobing Sun, Ji. Tongkai, Yue. Qiang and Feiya Xiong, 2011. IaaS Public Cloud Computing Platform Scheduling Model and Optimization Analysis. International Journal of Communications, Network and System Sciences, 12a(4): 803-811. DOI: 10.4236/ijcns.2011.432098.

Arfeen, M.A., K. Pawlikowski and A. Willig, 2011. A Framework for Resource Allocation Strategies in Cloud Computing Environment. IEEE Computer Software and Applications Conference Workshops (COMPSACW), pp: 261-266. DOI: 10.1109/COMPSACW.2011.52.

Armbrust, M., F. Armando, R. Griffith, D. Anthony and R. Joseph, 2009. Above the clouds: A Berkely view of cloud computing. University of California, Berkeley.

Bhilare, D.S. and Vivek Shrivastava, 2012. Algorithms to Improve Resource Utilization and Request Acceptance Rate in IaaS Cloud Scheduling.

International Journal of Advanced Networking and Applications. 03(05): 1367-1374.

Borja Sotomayor, S. Ruben, Montero, M. Ignacio, Lolrente and Ian Foster, 2009. An Open Source Solution for Virtual Infrastructure Management in Private and Hybrid Clouds. IEEE Transactions on Internet Computing. Special issue on Cloud Computing.

Chokhani, P. and G. Somani, 2013. Dynamic Resource Allocation using Auto-negotiation in Haizea. Contemporary Computing (IC3), IEEE Sixth International Conference, pp: 232-238.

Elghoneimy, E.O., Bouhall and H. Alnuweiri, 2012. Resource allocation and scheduling in cloud computing. IEEE International conference on Computing, Networking and Communications (ICNC), pp: 309-314. DOI: 10.1109/ICCNC.2012.6167434.

Gang Zhao., 2014. Cost-Aware Scheduling Algorithm Based on PSO in Cloud Computing Environment. International Journal of Grid and Distributed Computing, 7(1): 33-42. DOI: 10.14257/ijgdc.2014.7.1.

Guo, L., S. Zhao, S. Shen and C. Jiang, 2012. Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm. Journal of Networks, 7(3): 547-553.

Heba Kurdi, Ebtesam Aloboud, Sarah Alhassan and T. Ebtehal Alotaibi, 2014. An Algorithm for Handling Stravation and Resource Rejection in Public Clouds. 9th International Conference on Future Networks and Communications (FNC-2014). Elsevier Procedia Computer Science, 34: 242-248.DOI: 10.1016/j.procs.2014.07.018.

Jemina Priyadarshini, R., L. Arockiam, 2014. Performance Evaluation of Min-Min and Max-Min Algroithms for Job Scheduling in Federated Cloud, 99(18): 47-54.

Jiayin, Li., Qiu. Meikang, Niu. Jian-Wei, Yu. Chen and Zhong Ming, 2011. Adaptive Resource Allocation for Pre-empt able Jobs in Cloud Systems. 10th International Conference on Intelligent System Design and Application, pp: 31-36.

Jie Hui, Wei Zheng and Zhong You, 2014. Research for the Task Scheduling Algorithm Optimization based on Hybrid PSO and ACO for Cloud Computing. International Journal of Grid Distribution Computing, 7(5): 87-96. DOI: 10.14257/ijgdc.2014.7.5.08.

Kruekaew, B. and W. Kimpan, 2014. Virtual Machine Scheduling Management on Cloud Computing Using Artificial Bee Colony. Lecture Notes in Engineering and Computer Science: Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS), pp: 18-22.

Lin, C.T., 2013. Comparative Based Analysis of Scheduling Algorithms for Resource Management in Cloud Computing Environment. JCSE International Journal of Computer Science and Engineering, 1(1): 17-23.

Mohsen Amini Salehi, Bahman Javadi and Rajkumar Buyya, 2014. Resource Provisoning based on Preempting Virtual Machines in Distributed Systems. Concurrency and Computation: Practice and Experience, 26(2): 412-433. DOI: 10.1002/cpe.3004.

Shakkeera, L. and Latha Tamilselvan, 2013. Task Scheduling System for Multiple Workflows in Virtual Cloud Environment using Load Balancing Techniques. European Journal of Scientific Research. ISSN 1450-216X, 94(1): 120-131.

Sotomayor, B., R. Montero, I. Llorente and I. Foster, 2009. Resource leasing and the art of suspending virtual machines" in High Performance Computing and Communications. 11th IEEE International Conference, pp: 59-68. DOI: 10.1109/hpcc.2009.17.

Sotomayor, b.**,** R. Santiago Montero, I. Martín Llorente and I. Foster, 2009. Virtual Infrastructure Management in Private and Hybrid Cloud. IEEE Internet Computing, 13(5): 14-22.

Sotomayor, b., R.S. Montero, I.M. Llorente and I. Foster, 2009. Capacity Leasing in Cloud Systems using the OpenNebula Engine. Workshop on Cloud Computing and its Applications.

Sunilkumar Manvi, S. and Gopal Krishana Shyam, 2014. Resource Management for Infrastructure as a Service (IaaS) in cloud computing: A Survey. Journal of Network and Computer Applications 41 (Elsevier), pp: 424-440. DOI: 10.1016/j.jnca.2013.10.004.

Tharam Dillion, Chen Wu and Elizabeth Chang, 2010. Cloud Computing: Issues and Challenges. Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, pp: 27-33. ISBN:978-0-7695-4018-4. DOI:10.1109/AINA.2010.187.

Zhen Xiao, Weijia Song and Qi Chen, 2013. Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment. IEEE Transactions on Parallel and Distributed Systems, 24(6): 1107-1117. DOI:10.1109/TPDS.2012.283.