2017 International Conference on circuits Power and Computing Technologies [ICCPCT]

Experimental Analysis on Secured Agent Based Resource Matching and Virtual Grouping of Cloud Resources- An Integrated Approach

S.Lavanya
Assistant Professor,
Dept. of IT,
Sri Krishna College of Engineering and Technology,
Coimbatore, India.
lavanyamecsbit@gmail.com

Dr.N.M.Saravana Kumar
Prof. & Head,
Dept. of IT,
Vivekanandha College of Engineering for Women,
Tiruchengode, India.
Saravanakumaar2008@gmail.com

S.Thilagam
PG Scholar,
Dept. of IT,
Vivekanandha College of Engineering for Women,
Tiruchengode,
India.
thilak1993it@gmail.com

S.Sinduja
Assistant Professor,
Dept. of IT,
Vivekanandha College of Engineering for Women,
Tiruchengode,
India.
sindujanaga@gmail.com

***Abstract* - Resource allocation and utilization is the major part in cloud computing which enables customers to access the un-limited resources on demand and without any fixed upfront cost. The notion of cloud computing has not only reshaped the field of distributed systems but also fundamentally changed how businesses utilize computing today. An important issue faced in cloud in all the aspects is to match the needed resources to the consumers/users effectively i.e., with minimum or acceptable response time. A virtual grouping of resources has been done based on grouping of instance families and it resides in service providers. The virtual Machines (VM) offers the appropriate and more flexible way to configure and access the resources. A set of broker agents match consumers' requests to resources from providers. Empirical results show that mapper agents are doing well in matching requests to resources.**

Keywords - Resource allocation; Mapper agent; Request agent; Repair agent; Virtual Machines

I. INTRODUCTION

With the advancement in science and technology, the cloud computing has become a necessity now days when an enterprise plans to increase its' capacity or capabilities on the fly without investing on new infrastructure, training new personnel, buying new software licenses etc. It includes any subscription-based or pay-per-use service that extends the enterprise's existing IT capabilities, over the Internet in real-time. It is significantly necessary to utilize security controls and policies that protect sensitive data no matter where it exists, as point solutions by their nature they offer only limited visibility.

A Cloud computing system consists of a collection of interconnected and virtualized computers dynamically provisioned as one or more unified computing resource(s) through negotiation of service-level agreements (*SLAs*) between providers and consumers. Whereas huge corporations such as Amazon and Google can generate extra revenue by offering their occasionally under-utilized large-scale computing infrastructures (designed for their own peak demands), Cloud users (including *SMEs*) can benefit from reduced operating cost in maintaining their own computing infrastructure, rapid system provisioning, and expanded computing capabilities. Amazon EC2 cloud is considered here as example and its average response time is chosen for reference in proposed system [11]. This research proposes an agent-based mechanism for mapping and allocation of resources in a Cloud computing environment which reduces the average response time of the request.

In a Cloud-based business model, users pay service/resource providers for consumption of their computing capabilities similar to the way that basic utilities such as electricity, gas, and water that are charged, and it was noted that a *market-oriented approach* for managing Cloud resources is necessary for regulating their supply and demand through flexible and dynamic pricing.

In Cloud computing platforms, resources need to be dynamically (re-)configured and bundled (aggregated) via virtualization [1] and consumers' requirements can potentially vary over time and amendments may need to be accommodated. Supporting autonomous resource mapping and dealing with changing requests accentuate the need for Cloud resource management systems that are capable of continuously managing the resource reservation process by autonomously adjusting resource schedules and prices to accommodate dynamically changing resource demands. This research proposed a

978-1- 5090-4967- 7/17/$31.00 © 2017 IEEE

mechanism which solves the above mentioned issues in cloud environment.

Also, investigation on secure cloud computing is due to the extensive complexity of the cloud, it will be difficult to provide a holistic solution to securing the cloud, at present. Therefore, our goal is to make increment enhancements to securing the cloud that will ultimately result in a secure cloud. The major security challenge with cloud is that the owner of the data may not have control of where the data is placed. This is because if one wants to exploit the benefits of using cloud computing, one must also utilize the resource allocation and scheduling provided by clouds.

II. RELATED WORK

Market- based resource management [1] has been proposed by researchers to manage allocations of computing resources since it is effectively utilized in the field of economics to regulate supply and demand of limited goods. With Cloud computing emphasizing on a pay-per-use economic model, there is a high potential to apply market -based resource management techniques that justify the monetary return and opportunity cost of resource allocation according to consumer QoS expectations and baseline energy costs.

Detailed description of existing resource management systems in data centers [2,13,17] is yet to support Service Level Agreement (SLA)-oriented resource allocation, and thus need to be enhanced to realize cloud computing and utility computing. In addition, no work has been done to collectively incorporate customer-driven service management, computational risk management, and autonomic resource management into a market-based resource management system to target the rapidly changing enterprise requirements of Cloud computing. It presents vision, challenges, and architectural elements of SLA-oriented resource management and supports integration of market-based provisioning policies and virtualization technologies for flexible allocation of resources to applications.

It is noted that a Grid [3] as “a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed “autonomous” resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements", this paper adopts the position that many Cloud computing deployments depend on Grids, have autonomic characteristics (self management capabilities), and bill like utilities. The areas that are related to this work include: *agent-based* Grid resource discovery and Grid resource negotiation[14].

1) *Agent-based Grid resource discovery*: devised an algorithm for dynamically assembling agents that are capable of supplying information about distributed networked resources. Each agent periodically exchanges Grid resource information with other agents.

2) *Grid resource negotiation*: A relaxed criteria protocol has been proposed for Grid resource negotiation among market-driven agents. The negotiation agents take into consideration the market dynamics in a Grid and are programmed to slightly relax their negotiation criteria to enhance negotiation success rates.

Number of resource allocation methods effectively acts in cloud is Priority Based Allocation, optimization Task Scheduling Algorithm and Liner scheduling algorithm, etc. The optimization Task scheduling algorith [4] describes the fuzzy sets to model imprecise scheduling parameters and evaluates it to represent satisfaction grades of each objective. Genetic algorithms were also used to achieve task level scheduling in Hadoop Map Reduce.

The representation of task scheduling algorithm [6] based on load balancing in cloud computing. It describes two level task scheduling based on the load balancing. This scheduling does not meet consumer’s request but provides enormous amount of resource utilization. The implementation of Quality of Service (QoS) based Meta-Scheduler and Backfill strategy based light weight Virtual Machine Scheduler for dispatching jobs sounds better.

Hierarchical scheduling in [6] helps to achieve Service Level Agreement which results in quick response from the service provider. Quality of Service metric such as response time is achieved by executing deadline based jobs. It is estimated with the help of Task Scheduler by job completion time and the priority jobs then these are spawned from the remaining job.

An optimized algorithm [7] for task scheduling based on Activity Based Costing (ABC) assigns priority level for each task and uses cost drivers. ABC measures both cost of the object and performance of the activities In paper [8] analyzes and evaluation the various CPU scheduling in cloud environment using CloudSim the basic algorithm OS like FCFS, Priority Scheduling and Shortest Job First are tested under different conditions in which scheduling policy perform better[18].

III. SECURED AGENT BASED AND VIRTUAL GROUPING OF CLOUD RESOURCES

The cloud environment consists of a set of resource consumers and physical machines.VM ware which comprises of resource manager which oversees all resource pool for allocation of all capabilities such as network bandwidth, memory, CPU storage for all type of users upon request. Many instance runs in the VM ware like specific, generic, micro instances are kept in different data centers to process individually and secret key is sent to the user in order to provide security. The secret key generation can be explained in the further section.

The instance is running based on the instance family. Grouping of instances is meant to correctly categorize the resources and also for resource management. Consider resource pool R_1contains general purpose instance family with instance type, processor architecture, instance storage, physical processor, etc., R_2 contains compute optimized instance family with instance type, processor architecture, instance storage, physical processor, and so on which is taken from Amazon EC2. In proposed scheme, virtual instance pool is created in which the instance types are grouped from each instance family.

For example, R_1 refers to resource pool 1 which represents general purpose instance family that contains processor speed, storage and I/O operations. R_2 refers to resource pool 2 that represent compute optimized instance family which contains processor speed of storage, and I/O operations and so on that are fixed in Amazon EC2. This paper has been proposed to introduce the new concept called virtual grouping of resources in a separate pool such that if memory in GiB is less than or equal to 15 , vCPU memory needed is less than or equal to of 4GB, I/O operations is less than or equal to 2000 IOPS that are grouped under a single resource pool R_1 from combination of three instances such as general purpose, compute optimized and GPU instance family. Then the memory in GiB is greater than 15 and less than 60, vCPU memory needed is greater than 4 GB and less than 16, I/O operations are greater than 2000 IOPS and less than 6000 IOPS are grouped under another resource pool R_2 and so on with the consumer's acceptable cost. Here no matter of instance families but ultimate aim is meant to provide usage of small amount of resources to all the users.

Figure 1 shows that the users are grouped under clusters to match corresponding resources in the resource pools whereas R_1……R_n refer to resource pools where instance families are grouped. U_{ij} are the users accessing the particular pools mentioned as separate clusters. All the resource pools are managed by resource manager (RM).

With the above mentioned idea, 4 types of software agents namely Request agents, Mapper Agents, Weaker agents and Repair agents have been introduced for easy access of resource. These agents are multiple in numbers where in **Request agents** looks for consumer's requests, process all types of request <r_1...r_n> and saves in its cache for further reference whereas **Mapper agent** which comprises of many mappers in a system from <m_1....m_n> which maps the resources accurately to satisfy the user needs. When the new users/consumers are requesting for resources, service provider can easily identify the instance using agent based mechanism and assigns it to particular user where its instance services match. Data centers consists of <Instance 1.....Instance N>. All the agents are registered in Directory facilitator System (DFS). Example of grouped instances is shown in Table 1.

Figure 2 shows the sequence of steps to be followed during resource allocation.

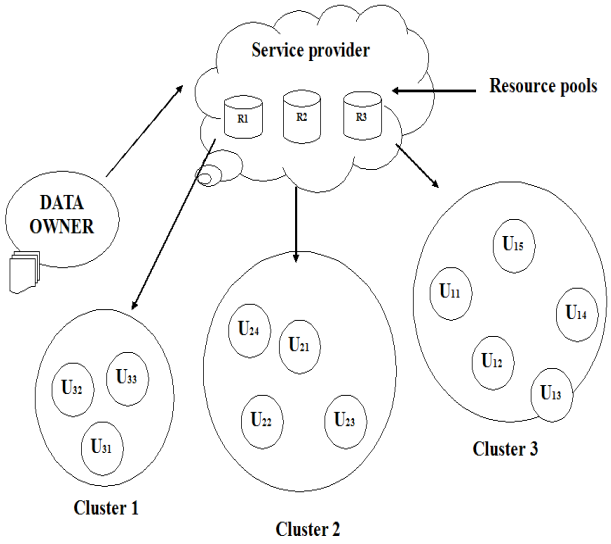


Fig. 1. Representation of Resource Pools

Table I. Grouped Resources in Resource Pools

S.No	Resource Pool R_1	Resource Pool R_2
1	a) memory in GiB <= 15 b) vCPU memory <=4GB, c) I/O operations <=2000 IOPS	a) memory in GiB > 15 and <= 60, b) vCPU memory needed > 4 GB and <= 16, c) I/O operations > 2000 IOPS and <= 6000 IOPS

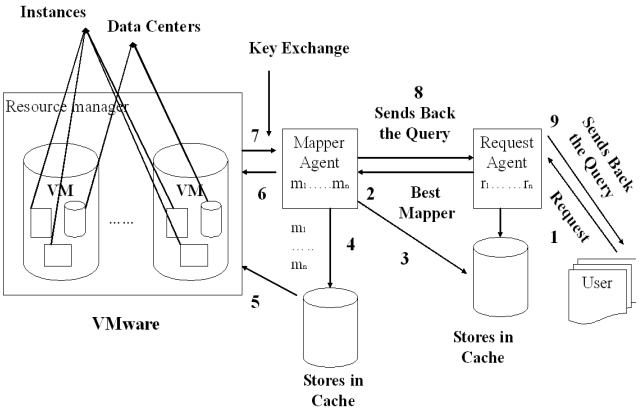


Fig. 2. Sequence of operations of resource allocation

Steps for resource allocation:

Step 1: Request agents look for consumers' request initially and stores at a requestor cache.

Step 2: Mapper agent looks at the requestor cache for further event to occur.

Step 3: Mapper sends the query to resource manager in which manager looks at the resource pool for resource allocation.

Step 4: Mapper agent maps resource on the basis of requirement and availability and then responses back to the requestor agent which then sends to the consumer upon successful allocation.

Fortunately by using multiple agents to handle multiple requests along with VMware allocation, it reduces the service time as well as waiting time and optimal fashion of resource allocation can be done in excellent manner. Here, the best mapper is the extra agent available which is used to find the mapper which is free next. In case of agent not found condition, error message is sent to DFS and it is handled by repair agent assigned by DFS as explained in next section.

Step 5:Resource advertisements in case of failure: Those agents who failed to do its job correctly, it is named as weaker agents. In case any agent fails, agent advertisements is carried out and the message sent to intermediate agent who is responsible for weaker agent to handle request and proceed further. There is an extended agent called as repairing agent(intermediate) which finds the fault in it and codes what does that error message contain and take steps to look over it. All agent discoveries under this cloud test bed contain cache for recording of all events during resource allocation**.**

IV. IDENTITY BASED VERIFICATION FOR SECURE COMMUNICATION

Security policy is ensured by using user identity, public-private key generation, signature generation and session key generation. User identity (ID) is a non-negative integer assigned uniquely to the user by the service provider. Each user is given long-term public and private keys. The provider randomly chooses a secret key and then computes and publishes the corresponding public key. The system uses the idea of RSA to construct a private-public key pair, where the provider calculates

(1) public key *(M, e),* where *M* is the product of any two large prime numbers, *a* and *b*, and e is the number prime with respect to *M* and

(2) private key *(a, b, d, φ(M))*, where *d* is the part of private key and is equal to e^{-1} *mod φ(M).*

The provider determines a primitive element *α* in *GF(a)* and *GF(b)*. Then it chooses a one-way hash function. Here, *(α, h())* is a public information where *h()* gives unique output for different input. Each provider provides *ID*s of the user under it and obtain the signature $S_{ij}$for each ID_{ij} of a user U_{ij} , where *i = 1, 2, 3, ...*, represents cluster and *j = 1, 2, or 3* represents the *user*.

$$S_{ij} = UID_i^d \bmod M \quad (1)$$

A. *Communication between the RM and the agent, agent and the user*

Let us know how to provide security duringcommunication between agents and the users, (i.e., agentand U_{ij}). Here the ID_{ij} are the user identities assigned to each user , public key (d) and private key(e), signatures(S_{ij}) of each U_{ij} are generated by service provider using RSA.

1) U_{ij} selects a random number R_{ij} and computes two public keys x_{ij} and y_{ij} as follows:

$$x_{ij} = S_{ij}.\alpha^{R_{ij}} \bmod M \quad (2)$$

$$y_{ij} = R_{ij}^e \bmod M \quad (3)$$

2) U_{ij} uses the identification number ID_i of CSPand finds P_{ij} by doing the operation one-way function of $h(x_{ij}, y_{ij}, ID_i)$, then computes

$$P_{ij} = S_{ij}.R_{ij}^{h(x_{ij}, y_{ij}, ID_i)} \bmod M \quad (4)$$

3) U_{ij} sends (ID_i, x_{ij}, y_{ij}) to agent.

Similarly, member agent selects the random number R_{kj} then computes x_{kj} , y_{kj} , and P_{kj} i.e.

$$P_{kj} = S_{kj}.R_{kj}^{h(x_{kj}, y_{kj}, ID_{ij})} \bmod M \quad (5)$$

and sends $(ID_{ij}, x_{kj}, y_{kj})$ to U_{ij}. U_{ij} and agent have to verify whether (ID_i, x_{ij}, y_{ij}) and $(ID_{ij}, x_{kj}, y_{kj})$ are sent from the user U_{ij} and the agent respectively. It is done by checking

$$P_{kj}^e = ID_i.y_{kj}^{h(x_{kj}, y_{kj}, ID_{ij})} \bmod M \quad (6)$$

Consider P_{kj} from Equation 6. From Equation 5,

$$P_{kj}^e = (ID_i^d \bmod M)^e .(R_{kj}^{h(x_{kj}, y_{kj}, ID_i)} \bmod M)^e$$

Mathematically, $(G^x \bmod n)^y = (G^y \bmod n)^x = G^{xy} \bmod n$ and $(G^x \bmod n) \bmod n = (G^x \bmod n)$ because n is a very large number. According to RSA, $d = e{-}1 \bmod \varphi(n)$ and $d * e = 1 \bmod \varphi(n) = 1$

$$P_{kj}^e = ID_i.y_{kj}^{h(x_{kj}, y_{kj}, ID_i)} \bmod M$$

which is similar to Equation 9 . Similarly, member verify at his end that

$$P_{kj}^e = ID_i.y_{ij}^{h(x_{ij}, y_{ij}, ID_i)} \bmod M \quad (7)$$

The communicating members compute a secret session key (*SK*). The computation of *SK*s is as follows: Consider the communication between user U_{ij} and agent. They compute the secret *SK*s SK_{ij} (user's session key) and *SK (*agent's session key) respectively as follows :

$$SK_{ij} = (\frac{x_{kj}^{e}}{ID_i})_{kj}^{R} \bmod M \quad (8)$$

$$SK = (\frac{x_{ij}^{e}}{UID_{ij}})_{ij}^{R} \bmod M \quad (9)$$

$SK_{i,j}$ and $SK_{k,j}$are the same for the communicating members. Hence,

$$SK_{ij} = SK = \alpha^{e \oplus R1 \oplus R2} \bmod M \quad (10)$$

Since, M is a very large, the above equation can be written as

$$SK_{ij} = SK = \alpha^{e \oplus R1 \oplus R2} \quad (11)$$

Using these session keys, they can communicate successfully with each other. If the Session key is same for both (i.e., agent and the user), the user and the agent is authenticated and is trusted. It ensures data integrity and authenticity. Session key is used to encrypt/decrypt the data between the agent and the user.

V. EXPERIMENTAL RESULT

Figure 3 shows the list of instance families taken for experiments which is for grouping and allocation of resources.

Table II. Range of Parameters

Parameter	Minimum Number	Maximum number
Instance pool Created	3	9
Group/Clusters created	3	5
No. of Instances	3	12
No. of Agents	4	n
Operating System	Linux	
Server	Apache Tomcat 6	

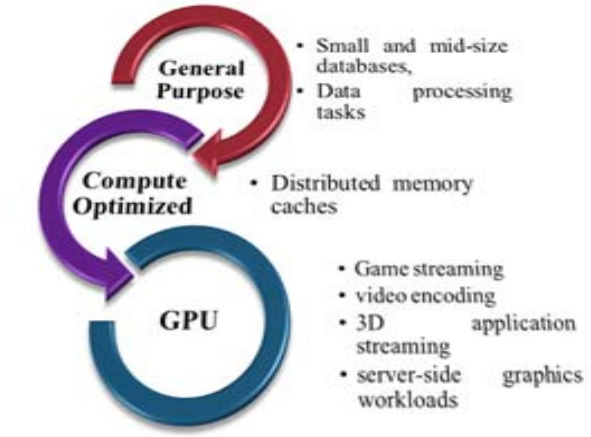


Fig. 3. Instance families used for experiments

A. Experimental Results

Figure 4 shows the implementation of instance running virtually.

Figure 5 shows that matching of resources is done and allocation of resources is being completed successfully on the basis of resource availability through resource manager.

Figure 6 indicates that, resources were retrieved from the cloud based on user requirement and user can make use of resources.

Figure 7 indicates how users are initiating their request where the instances are running and other user logs are being projected.

Figure 8 shows the server log in which consists of information like who are all accessing the resources, on what basis, allocated resource details, available resources, etc.

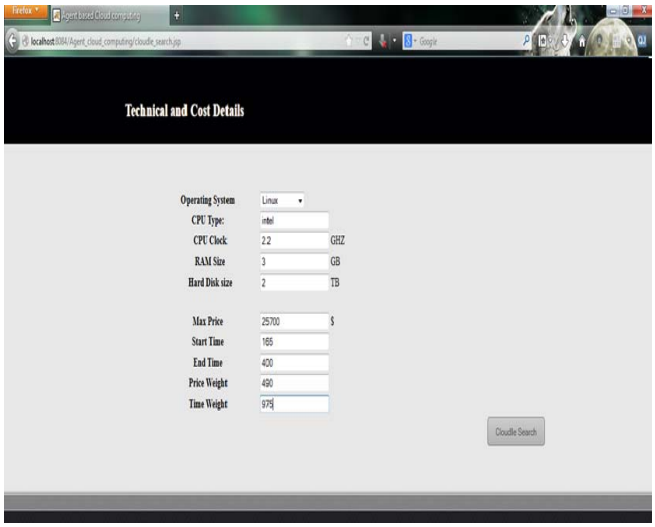


Fig. 4. Instance Cretion setup

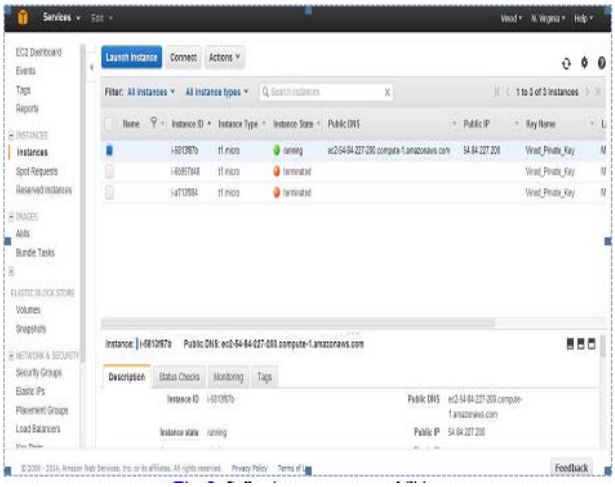


Fig. 5. Collecting resource capabilities

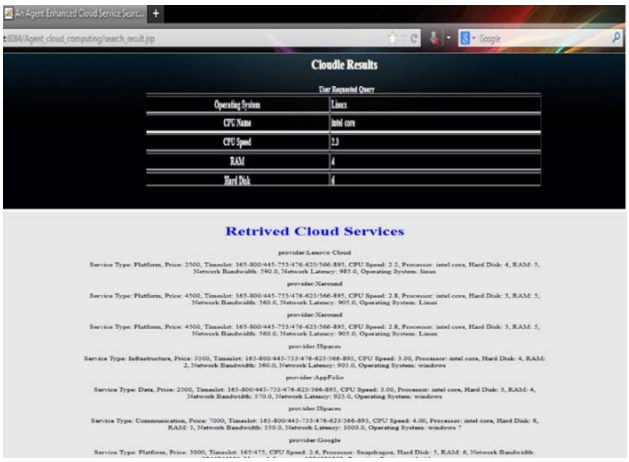


Fig. 6. No. of services retrieved

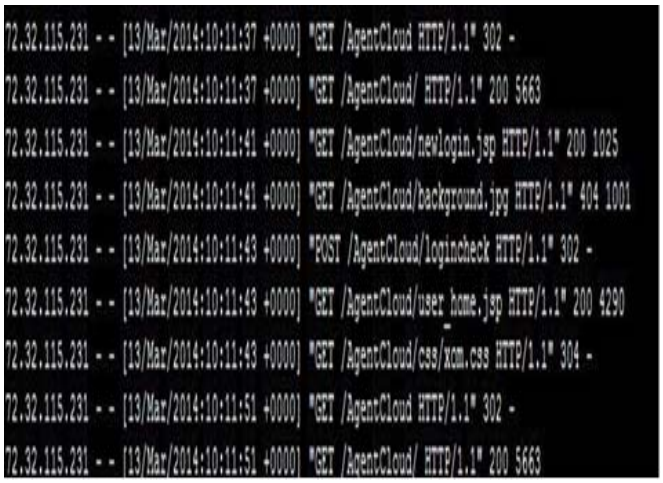


Fig. 7. User log

```
Mar 13, 2014 9:50:02 AM org.apache.catalina.core.AprLifecycleListener init
INFO: The APR based Apache Tomcat Native library which allows optimal performance in production environ
.6.0-openjdk-1.6.0.0.x86_64/jre/lib/amd64/server:/usr/lib/jvm/java-1.6.0-openjdk-1.6.0.0.x86_64/jre/lib
amd64:/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib
Mar 13, 2014 9:50:03 AM org.apache.coyote.http11.Http11Protocol init
INFO: Initializing Coyote HTTP/1.1 on http-8080
Mar 13, 2014 9:50:03 AM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 2692 ms
Mar 13, 2014 9:50:03 AM org.apache.catalina.core.StandardService start
INFO: Starting service Catalina
Mar 13, 2014 9:50:03 AM org.apache.catalina.core.StandardEngine start
INFO: Starting Servlet Engine: Apache Tomcat/6.0.37
Mar 13, 2014 9:50:03 AM org.apache.catalina.startup.HostConfig deployDescriptor
INFO: Deploying configuration descriptor host-manager.xml
Mar 13, 2014 9:50:04 AM org.apache.catalina.startup.HostConfig deployDescriptor
INFO: Deploying configuration descriptor AgentCloud.xml
Mar 13, 2014 9:50:04 AM org.apache.catalina.startup.HostConfig deployDescriptor
INFO: Deploying configuration descriptor manager.xml
Mar 13, 2014 9:50:05 AM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory docs
Mar 13, 2014 9:50:05 AM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory examples
Mar 13, 2014 9:50:05 AM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory sample
Mar 13, 2014 9:50:05 AM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory ROOT
Mar 13, 2014 9:50:06 AM org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on http-8080
Mar 13, 2014 9:50:06 AM org.apache.jk.common.ChannelSocket init
INFO: JK: ajp13 listening on /0.0.0.0:8009
Mar 13, 2014 9:50:06 AM org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=0/35  config=null
Mar 13, 2014 9:50:06 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in 2728 ms
Database_Connected
connected
Database_Connected
connected
```

Fig. 8.Server access log

Table III. User request Vs Job allocation time in various approaches

No. of requests	Priority based Allocation (in seconds)	Task scheduling Algorithm (in seconds)	Average Job Allocation time in Agent Based Allocation (in seconds)
20	0.504	0.502	0.401
40	0.685	0.663	0.582
60	0.987	0.739	0.693
70	1.200	0.811	0.727

B. Experimental Analysis

Different set of jobs simulation is done in 10 runs. In each run of simulation, a set of 85 different service requests (i.e. jobs), and each service request is composed of up to 12 sub-tasks. Four clouds are considered in the simulation. All 85 service requests will be submitted to random clouds at arbitrary arrival time. The parameters in Table 1 are set in simulation randomly according to grouped resources. Since we focus only on effective resource allocation, we do our simulations locally without implementing in any exiting cloud system or using VM interface API. The following graphs and parameters specify the effectiveness of the idea. Table 3 and Figure 9 explain the average time required to allocate the jobs (in seconds) in various approaches while number of user's request is increasing.

Table 4 and Figure 10 explain the computation cost of proposed secure scheme for computing different keys when number of users is increasing in number. Table 4 clearly illustrates that the computational cost of keys generation for communication between the RM, user and the agent is only one.

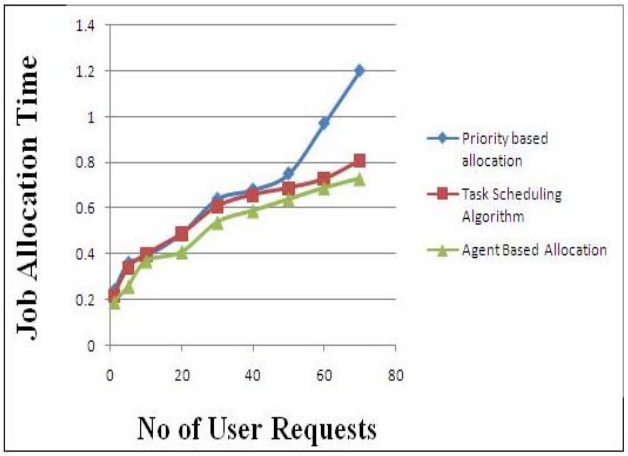


Fig. 9.Average number of job allocation time Vs number of user request

Table IV. Number of Users Vs Computation Cost

S.No	No. of Users	Computation Cost
1	20	O(1)
2	40	O(1)
3	60	O(1)
4	70	O(1)

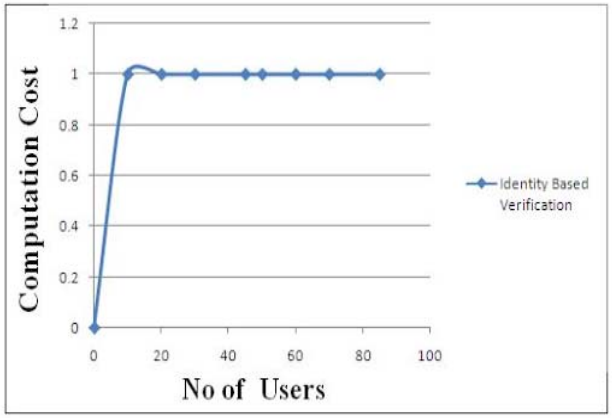


Fig. 10.Computation Cost Vs Number of User

VI. CONCLUSION

While scheduling and management of VMs in a cloud environment is very important, one must realize what is actually being scheduled. In a normal Cloud environment like the Amazon's EC2, full Operating System VMs are scheduled, often to carry out specific tasks. In case of proposed system, use of grouping mechanism to reduce search time and agent mechanisms for resource allocation provides a better way for optimal service time .Security and secure communication establishment is ensured with the usage of session key. Future enhancement towards advanced optimization techniques for quick results and bench mark the recent study with many computing like grid, pervasive, utility computing, etc. The experimental result shows that the proposed scheme reduces response time by 0.28 seconds in Amazon cloud EC2.

REFERENCES

[1] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic Models for R esource Management and Scheduling in Grid Computing", Concurrency and Computation: Practice and Experience, Vol.14, pp.1507- 1542, Nov.Dec. 2002.

[2] Abirami S P and Shalini Ramanathan, "Linear Scheduling Strategy For Resource Allcocation In Cloud Environment", International Journal on cloud computing: Services and architecture, Vol. 2, No 1,pp: 9-17 ,Feb 2012.

[3] Kwang Mong Sim, "Agent Based Cloud Computing", IEEE Transaction on Services Computing, Vol. 5, No.4, 2012.

[4] Savitha.P and J Geetha Reddy, "A Review Work On Task Scheduling In Cloud Computing Using Genetic Algorithm", International Journal Of Scientific & Technology Research, Vol.2, No.8, pp.241-245, 2013.

[5] Dr. Sudha Sadhasivam, R. Jayarani, Dr. N. Nagaveni and R. Vasanth Ram, "Design and Implementation of an efficient Two - level Scheduler for Cloud Computing Environment", In Proceedings of International Conference on Advances in Recent Technologies in Communication and Computing, 2009.

[6] Rajkumar Rajavel and Mala T "Achieving Service Level Agreement in CloudEnvironment using Job Prioritization in Hierarchical Scheduling", In Proceeding of International Conference on Information System Design and Intelligent Application, vol .132, pp 547-554, 2012.

[7] Q. Cao, W. Gong and Z. Wei, "An Optimized Algorithm for Task Scheduling Based On Activity Based Costing in Cloud Computing", In Proceedings of Third International Conference on Bioinformatics and Biomedical Engineering, pp. 1-3, 2009.

[8] Monica Gahlawat and Priyanka Sharma (2013) " Analysis and Performance Assessment of CPU Scheduling Algorithm in Cloud Sim" International Journal of Applied Information System(IJAIS)-1SSN: 2249 -0868 Foundation of Computer Science FCS, New York, USA Vol.5, No. 9,July 2013.

[9] Zhen Xiao, Weijia Song and Qi Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment", IEEE Transactions on Parallel and Distributed Systems , Vol. 24, No. 6, pp: 1107-1117, June 2013.

[10] Sharkh, M.A. ; Jammal, M. ; Shami and A. ; Ouda, "A. Resource allocation in a network-based cloud computing environment: design challenges", Communications Magazine, IEEE ,Vol.51 ,No.11, pp. 46-52, Nov 2013.

[11] V.Vinothina, Dr.R.Sridaran and Dr.PadmavathiGanapathi, "A Survey on Resource Allocation Strategies in Cloud Computing", International Journal of Advanced Computer Science and Applications,Vol.3, No.6, pp.97-104, 2012.

[12] G. Lee, N. Tolia, P. Ranganathan, and R. H. Katz, "Topology-Aware Resource Allocation for Data Intensive Workloads", In Proceedings of the Asia-Pacific Workshop on Systems (APSys) , 2010.

[13] Bharti Ratan Madnani and Sreedevi N, "Attribute Based Encryption For Scalable And Secure Sharing Of Medical Records In Cloud Computing Design And Implementation", International Journal of Innovative Research in Computer and Communication Engineering, Vol. 1,No.3, pp 523-530, 2013.

[14] Pelin Angin and Bharat Bhargava, "An Agent Based Optimization Framework For Mobile – Cloud Computing", Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, Vol. 4,No 2, pp 1-17,2014.

[15] S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubramanian, K. Talwar, and L. Uyeda, "Validating Heuristics for Virtual Machines Consolidation", Technical Report MSR-TR-2011-9, Microsoft Research, 2011.

[16] Amazon. Elastic Compute Cloud (EC2). http://www.amazon.com/ec2/.

[17] A. García García et al, "SLA-driven dynamic cloud resource management", Future Generation Computer Systems, Vol.31, pp.1–11, 2014.

[18] Medhat A. Tawfeek, Ashraf El-Sisi, Arabi E. Keshk and Fawzy A. Torkey "Cloud Task Scheduling Based on Ant Colony Optimization" In Proceeding of IEEE International Conference on ComputerEngineering & Systems (ICCES), 2013.