2015 IEEE International Conference on Communication Networks (ICCN)

Preemptable Priority Based Dynamic Resource Allocation in Cloud Computing with Fault Tolerance

Shubhakankshi Goutam
M.Tech. Scholar,
ITM University Gwalior,
Madhya Pradesh, India
shubhakankshi@gmail.com

Arun Kumar Yadav
Associate Professor
ITM University Gwalior,
Madhya Pradesh, India
arun26977@rediffmail.com

***Abstract*— Today, cloud computing serves as a request response model, where a client makes request for various available services on "pay as you go basis". Cloud computing offers a dynamic flexible resource allocation phenomenon. For reliable and guaranteed services there must be a scheduling mechanism that all resources are efficiently allocated to satisfy the customer's request. Cloud services are based on scalability, availability, security and fault tolerance features. Service provisioning in cloud is based on SLA. Service level agreement is the terms of cloud provider's contracts with customers to define the level(s) of service being sold in plain language terms. QoS (quality of service) plays important role in cloud environment. Resource scheduling and service deployment is done by considering multiple SLA parameters like CPU requirement, network bandwidth, memory and storage. In this paper we propose an algorithm which perform resource preemption from low priority task to high priority task and advanced reservation for resources considering multiple SLA parameters for deploying service. This algorithm is also effective for fault tolerance mechanism.**

Keywords- Cloud computing,SLA, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Resource management, Software as a Service (SaaS), Virtual machine, Virtualization.

I. INTRODUCTION

Cloud computing is "providing computing resources and applications over the internet" using pay as you go model. It is also known as internet computing, here a pool of resources such as memory, processor, network and bandwidth, virtually distributed across the internet. If a customer wants to use the services of cloud provider then it has to pay cost according to services using real time as per requirements. Cloud computing provides globalize sharing of resources and unlimited storage capacities.

As we know that any number of customers can make requests to cloud provider, if SLA based agreement takes place that means cloud provider are able to attain the corresponding request from the user, this is done by efficiently scheduling of resources and deploying applications on proper VMs. Resource scheduling means multiplexing of several user requests on same physical structure. In this time there exists more work already done on scheduling of resources in clouds this approach is based on global resource deployment by considering one SLA objective such as cost of execution, time of execution, minimum resources etc.

When a request is submitted by the client, it's firstly partitioned into several subtasks. Now there are four main responsibility of cloud scheduler, 1) find appropriate manner or order to execute a task, 2) find appropriate resource allocation schedule for task, 3) the scheduler should be fault tolerance to schedule overheads and terminate the task, 4) find a manner for migration of tasks in an effective manner. Using the resource allocation phenomenon these problems can be solve.

The scheduling can be two types, global scheduling and local scheduling. Local scheduling is a type of scheduling where localize resources are use to satisfy the user's request within one single cloud. Global scheduling is where all the resources from multiple clouds are treated as one single unit to complete the user's request. Typically efficient provisioning required two distinct processes, 1) initial static planning – local scheduling where all the VMs are mapped to physical resources. 2) Dynamic resource provisioning-create new VMs, migration on VMs, dynamically response according to workload. Here step 1 is set up stage, it is generally performed at the time of set up a cloud, and when maintenance is done by the source. Whether step 2 run repeatedly at the allocation time. There are various challenges arise in this area for researchers such as scalability, multitenency, security, dynamic resource allocation and fault tolerance.

In this paper we focus on dynamic resource provisioning, we present a scheduling heuristic considering multiple SLA objectives, such as amount required CPU, network bandwidth, and cost for deploying applications in clouds. The scheduling present a flexible on demand resource allocation strategy included advanced reservation and preempt-able mechanism for resources. Our proposed algorithm dynamically responds to requested resource for the task. First it's locally checks for the availability of resource; if resource is free then it deployed new VMs to current task, If resource is not available then it's create new VM from globally available resource; if global resources are not available then it will check for resource if it's preempt-

978-1-5090-0051-7/15/$31.00©2015 IEEE
DOI 10.1109/ICCN.2015.54

able then it's migrate processes otherwise put the task into waiting list and apply advanced reservation scheme.

Organization of papers is as follows: In **section 2** we discuss research works related to this topic, in **section 3** models for resource allocation and task scheduling in Iaas cloud computing system and consist the previous algorithms that is being used in our algorithm, **section 4** discusses the proposed method, **section 5** shows the simulation results, **section 6** conclude the paper and future work.

II. RELATED WORK

Jiayin Li [1] presents a resource optimization mechanism in heterogeneous IaaS federate multi cloud systems, that enable preemptable task scheduling with resource allotment model, cloud system model, local mapping and energy consumption, and application model. It is suitable for autonomic future in cloud and VMs. They proposed online dynamic algorithms for resource allocation and task scheduling. In proposed cloud resource phenomenal every data center has a manager server, the communication and resource allotment scheme works between various servers of each data center for share workloads among multiple data servers. The workload sharing makes a large resource pool of flexible and cheaper resources to resource allocation.

S. Pandey [2] presents a scheduling heuristic for optimization of the cost of task resources mapping based on solution of the practical swarm optimization technique, PSO is an intelligent algorithm influence by social behavior of animals such as a flock of birds finding a source of food or a school of fish protecting themselves from a predator. A particle in PSO is analogous to a bird or a fish flying through a search space the movement of every particle is consist a velocity The movement of each particle is co-ordinate by a velocity which has both magnitude and direction. Each particle position at any instance of time is influenced by its best position and the position of the best particle in a problem space. The performance of a particle is measured by a fitness value, which is problem specific. A fitness value is use in this algorithm which gives the best position to particle. PSO is a scheduling heuristic for dynamic scheduling application.

Vincent C. Emeakaroha [3] presents a scheduling heuristic aims on allocation on VMs based on SLA level terms and conditions. It concentrates on physical resource deployment based on resource availabilities, so the SLA violation reduces using this approach. It provides an integrated load balancer for high and efficient resource utilization in cloud computing environment. The algorithm provides local load balancing strategy and global resource allocation phenomena for better resource utilization in cloud environment.

Chandrashekhar S. Pawar [4] represents an algorithm for priority based dynamic resource allocation strategy in cloud environment, it provides modified scheduling heuristic in [3] for executing highest priority task that is consider as AR(advanced reservation) by preempting best effort task. This algorithm gives good performance in multi cloud environment. It can run on different set of jobs over multiple clouds. Various requests can be run in the AR modes to the SLA objectives.

Zhen Xiao [5] presents design and implementation of an automated resource management system that can avoid overload in the system while minimizing numbers of servers being used, it introduce the concept of skewness measure the convent utilization of the server and improve utilization of servers of multidimensional resource constraints. [4] Design a load prediction algorithm that can guess the future resource usages of the application without internal details.

Dorian Minarolli and Bernd Freisleben [6] represents a VM resource allocation in cloud computing via multi agent fuzzy control, it focused on line grained dynamic resource allocation of VM locally on each physical machine of a cloud provider and consider memory and CPU as resource that can be managed. Fuzzy control is used to minimize a global utility function as n a hill climbing heuristic implemented over fuzzy rules. The problem considers is how to resource of a cloud provider should be reallocated to VM dynamically where workload changes to keep the performance according to SLA's.

III. SCHEDULING STRATEGY AND DESIGNING ISSUES

In this section we are proposing diagram based on cloud environment and scheduling heuristic. Our proposed algorithm is generally based on SLA based resource provisioning and online adaptive scheduling for advanced preempt-able task execution. Two basic steps are required for effective utilization of cloud resources that meet the SLA objectives.

A. Resource allocation and deployment of application

There are three kinds of layers, that is combining to used in resource provisioning to service the request of user's. It's generally known as service models in cloud computing. IaaS, SaaS, and PaaS.

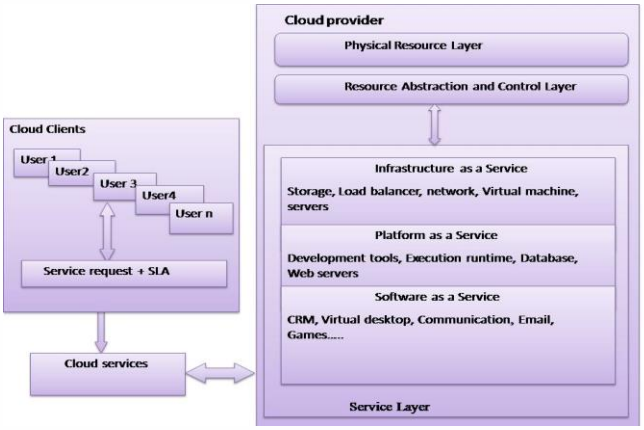


Fig.1. Cloud Services Layers

In “Fig. 1” shows how to cloud client connects with cloud provider to use various services. The SLA terms and condition should satisfy first. There are three kinds of layers in the given diagram: service layer, control layer and physical resource layers. Control layer controls the resource allocation and virtualization, resource layer contains physical resources within a cloud, service layer contains three type of services IaaS, PaaS and SaaS.

SaaS (software as a service) delivers applications and software on user’s demand that are managed by third party. Every step including applications, virtualization, runtime, Qos, server, storage managed by the vendor.

PaaS (plateform as a service) delivers plateform for application development, Using the Paas services a plateform is provided according to user’s request to development purpose.

IaaS (infrastructure as a service) provied computing resources on user’s demand such as physical or VMs, file based storage, load balancers, and firewall.

The cloud provisioning and application deployment model [3] is shown in fig[2]. That combines three types of service models. in each layer, consist their responsibilities for resource provisioning to meet the SLA objectives. As in fig[2] customer make their requests to service portal (step1). Service portal passes their request to processing and management component to check the validation (step2). If request is validate then it should be forworded to scheduler and load balancer for appropriate resource provisioning(step3). Load balancer invoked the provisioning engine of Paas layer for select the appropriateVM, and balances the service provisioning among running virtual machines(step4). Virtualization layer is use to manage the virtual machine and interaction with physical resources using provision engine of Iaas layer (step6).

Low level resource matrics and physical resource in Iaas layer are monitor by the LoM2HiS framework (step6). If SLA violation occurs, reactive actions provide by the knowledge database techniques in FoSII (step7). The requested service status and the SLA information communicated back with the service portal (step 8).

Provisioning can be done at the single layer alone. But our approach and algorithm aims to provied an integrated resource scheduling strategy. So we consider the three layers. IaaS layer is responsible for physical resources management. PaaS layer have a responsibility for deploye and managed VM that is mapped with physical resource considering the agreed SLA’s with the customers.

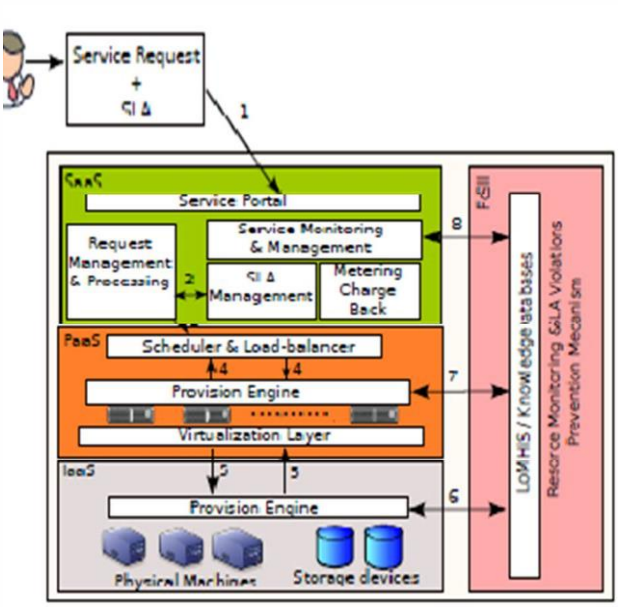


Fig. 2. Cloud Provisioning and Deployment model

In Figure 3 it’s shows the local and global resource scheduling between multiple clouds, as we know in cloud scheduling there are n numbers of users that can request any time for resources to cloud provider, if all the resources of current cloud is busy at a time and new request arrives have high priority then cloud provider have to put the request in waiting list, so here is a solution provided in this algorithm to effectively schedule and distribute user’s request across the multiple clouds, and preempt the resource from low priority task to high priority task. Our given algorithm is also capable to handle fault tolerance in cloud system.

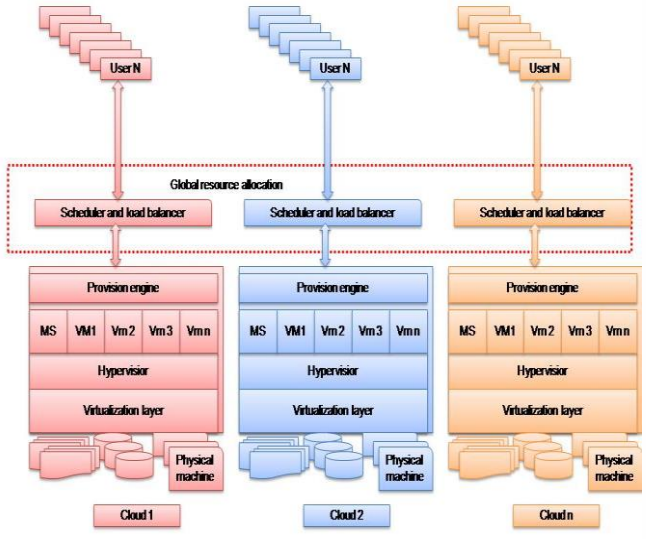


Fig. 3. Resource provisioning and scheduling between multiple clouds

B. *Scheduling heuristic description*

When a scheduler receives a user's service request, it will first partition that task request into several subtasks and form a DAG. This process is called as static resource allocation. In [1] authors proposed greedy algorithms to generate static planning of allocation: CLS (cloud list scheduling algorithm)

1. *Cloud list scheduling (CLS)*

This CLS is similar to CPNT [14]. The definitions used for listing the task are provided as follow. The earliest start time (EST) and the latest start time (LST) of a task are shown as in (1) and (2).The entry-tasks have EST equals to 0. And The LST of Exit-tasks equal to their EST.

$$EST(v_i) = \max_{v_m \in pred(v_i)} \{EST(v_m) + AT(v_m)\} \ldots\ldots (1)$$

$$LST(v_i) = \max_{v_m \in succ(v_i)} \{LST(v_m) - AT(v_m)\} \ldots\ldots (2)$$

As the cloud system concerned in [11] is heterogeneous the execution time of each task on VMs of different clouds are not the same. $AT(V_i)$ is the average execution time of task V_i. The critical node (CN) is a set of vertices having equal EST and LST in the DAG. Algorithm 1 shows a function forming a task list based on the priorities.

Algorithm 1 forming a task list based on priorities

Require (input): A DAG, Average execution time
AT of every task in the DAG
Ensure (output): A list of task P based on priorities
1. The EST is calculated for every task
2. The LST is calculated for every task
3. The T_p and B_p of every task are calculated
4. Empty list P and stack S, and pull all task in the list of task U
5. Push the CN task into stack S in decreasing order of their LST
6. While the stack S is not empty do
7. If top(S) has un-stacked immediate predecessors then
8. S ← the immediate predecessor with least LST
9. Else
10. P ← top(S)
11. Pop top(S)
12. End if
13. End while

Once the above algorithm 1 form the list of task according there priority, we can allocate resources to tasks in the order of formed list. When all the predecessor tasks of the assigned task are finished and cloud resources allocated to them are available, the assigned task will start its execution. This task is removed from the list after its assignment. This procedure is repeats until the list is empty.

IV. SCHEDULING ALGORITHM

In this section propose an algorithm to handle the priority request from the customer, and provide the advanced reservation and preemption over the resources, it is a modified version of previous algorithm in [4]. Here the highest priority of task defines over AR task and task related to highest cost paying by the customers.

Algorithm 2 Priority based scheduling algorithm

1. Input: AP(R, AR)
2. //call Algorithm 1 to form the list of task based on Priorities
3. Get global Available VM List and
   Get global Used VM List and also
   Available Resource List from each cloud scheduler
4. // find the appropriate VM List from each cloud Scheduler
5. If AP(R, AR)! = ϕ then
6. Get local available VM list and
   Local available resource list and
   Get local used VM list
7. Check for appropriate resource
   If,
   Found in local available list
   Then
   Start new VM Instance
   Add VM to used VM List
   Deploy service on new VM
   Deployed=true
8. Else if,
   Global Resource Able To Host Extra VM then
   Start new VM Instance
   Add VM to Available VM List
   Deploy service on new VM
   Deployed=true
9. Else if,
   If R= "high cost task" or R= "Advance reservation" && available VM<= requested VM
   // call algorithm 3 for preemption and advanced reservation
10 Update global and local available and used VM list
11 Else
   Queue service Request until
   Queue Time > waiting Time
12 Deployed=false
13 End if
14 If deployed then
15 return successful
16 terminate
17 Else
18 return failure
19 terminate

In algorithm [2] R is customer's service request, A is application data, S is SLA based terms and condition. These provided as input to the scheduler (step 1). When the request for a service send to cloud scheduler then scheduler divides it in many sub tasks as per their dependencies, for this purpose algorithm 1 is called. Algorithm 1 is also used to form a list of tasks based on their dependencies (step 2). In (step 3) scheduler get global available VM list, and the entire resources list, that is available for deployment user's services from the each cloud. Used VM list is also provided to add deployed VM information. In (step 4-5) it uses the SLA terms to find a list of appropriate VMs that is capable to provisioning the request service R. In (step 6) scheduler get all the local information and lists, once it have all information then load balancer locally decides which VM is allocate to service request (step 7). When there are no VM and requested resource is available at that time then scheduler globally checks for free resources if any resource is free globally then it deploy service on the resource by creating a new instance of VM (step 8). Else if there are no extra resources available locally and globally then it checks for task priority , if task is high cost price task or it has advanced reservation then scheduler runs algorithm (3) (step 9). In (step 10) scheduler updates their list if any changes occurs, during the resource allocation. In any other cases provision request is add to queue of waiting list until the VM with desire resource is get free (step 11). If after a certain period of time the service request is schedule and deployed then scheduler returns as successful deployment status otherwise it returns failure (step 14-19) to admin.

Algorithm 3 Advanced reservations and preemption based cloud min-min algorithm-

Input: A set of tasks, m different clouds ETM matrix
Output: A schedule generated by CMMS or preemption report

1. For a mappable task set P
2. While there are tasks not assigned do
3. Update mappable task set P
4. For I = task $V_i \in P$ do
5. Send task check requests of V_i to all other Cloud schedulers
6. Receive the earliest resource available time Response and list of task with their priorities Form all other cloud scheduler
7. Find the cloud $C_{min}(V_i)$ giving the earliest finish time of V_i, assuming no other task preempts V_i
8. If Resource R_i= "preemptable" Update circular queue for particular resource add new task V_i Cloud scheduler picks the task from the ready queue and set a timer to interrupt it after 2 time slice and dispatch it.
   A. If task brust time<=2
      1. Process while leave the resource after the completion
      2. Process next process in ready queue
   B. Else
      If task burst time>2
      1. Timer= -1
      2. Execute process and put it at the tail of circular queue
      3. Repeat step 1,2,3 until circular queue="empty"
   C. Deployment=true for the task
9. End for
10. Else if
11. Find the task-cloud pair (V_k, $C_{min}(V_k)$) with earliest finish time in the pairs generated in for loop
12. Assign task V_k to cloud $D_{min}(V_k)$
13. Remove V_k from P
14. Update the mappbale task set P
15. else if
    All the running task on cloud ="AR" tasks
    Then
    1. Find the resource R_i with earliest finish task V_j and advanced reservation='null'
    2. Make advanced reservation on particular resource R_i by task V_j
    3. After completion the previous the task V_j, R_i will be assigned to task V_i
16. End while

A cloud scheduler records execution schedule of all resources using a slot. A mappable task set is assigned to the algorithm, A mappable task set is a set whose predecessors task are allocated to VM and corresponding resource .If there is a high priority task (high cost) then it should be given advanced reservation according to given algorithm. If a task did not found appropriate resource in free condition and scheduler has a high priority task then this algorithm works in three steps.

1) First it will check for earliest resource available time from all the clouds then it's check current requested resource is preemptable or not , if yes then switching perform between the tasks on same resource using a time quantum. So the deployment status of current task is true and all the succeeding tasks can be successfully deployed on the VM.

2) If the requested resource is not preemptable and an AR task is assigned to a cloud, first resource availability in this cloud will be checked by cloud scheduler. Since best-effort task can be preempted by AR task, the only case when most of resources are reserved by some other AR task. Hence there are not enough resources left for this AR task in the required time slot. If the AR task is not rejected, which means there are enough resources available for the task, a set of required VMs are selected arbitrarily.

3) If all the tasks running on cloud are AR task then our algorithm gives advanced reservation on the resource of earliest finish task.

In preemptable priority if resource is preemptable then it just checks for nearest resource and assigns it to task then perform switching between tasks. When a task completes then it remove them from circular queue and return the

result. In advanced reservation, if a resource is nonpreemptable then scheduler just sends task checks request to all other cloud provider and receive the earliest available time of corresponding resource and then, the manager server of this cloud will first check the resource availability in this cloud. Since AR tasks can preempt best-effort tasks, if the resources are reserved by some other AR tasks at the required time, then, AR task will capture the resource by advanced reservation when resource get free.

Algorithm 4 Algorithm for fault tolerance

1. If current running tasks T_i= {T_1,T_2,……..,T_n}
2. Current resources assign to task T_n= {R_1, R_2,…….,R_n }
3. "ERROR OCCURE"
4. Resource status of R_i="FAILED"
5. Then
6. Do assign highest priority to T_i
7. T_i (priority)="highest"
8. Run algorithm 2
9. If deployed then
10. Deploy="true"
11. Return="successful"
12. Else
13. Suspend all the succeeding tasks
14. put task {T_i, T_{i+1} ,T_{i+2} ……,T_n} into waiting list
15. Deploy="false"
16. Return="unsuccessful"
17. End

Algorithm 4 is used for fault tolerance mechanism. All the service request and their deployment on VM is done in dynamic environment, in cloud computing there are still chances that resource get failed during the execution of tasks. Then there is a responsibility to cloud scheduler that mark that resource as failed status, and immediately migrate the task on new VM. For this purpose algorithm 2 can be a better option, because it efficiently allocates new resource.
In algorithm 4 current running task=T_i, current resource assign to task T_i is R_i (in step 1-2). If any resource casus failed due to hardware failure (step3) then scheduler assigns highest priority to that task T_i (step 6) and run algorithm 2 (step 7). In (step8-10) if deployment of corresponding task is done using algorithm 2 then it returns successful otherwise scheduler suspends all the succeeding tasks and put it into waiting list.

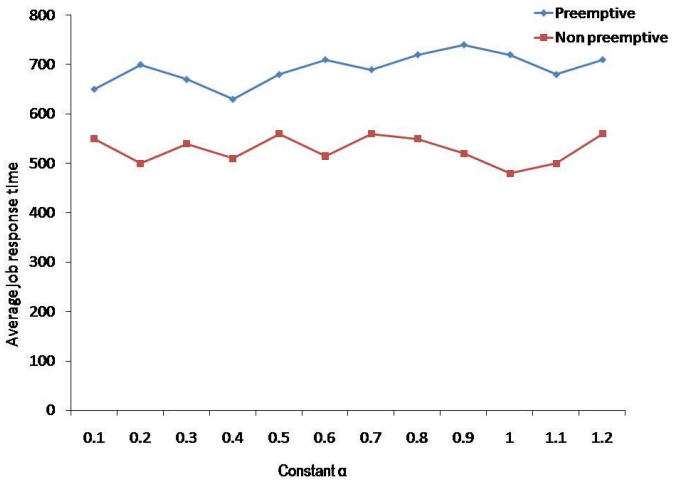
V. SIMULATION RESULT

In this section we discuss the experiment result of our given algorithm. We evaluate the performance of our given heuristic through simulation, by using different set of tasks in 10 runs. We experiment with a set of 60 different service requests each service request is composed in 10 to 15 subtasks. We consider 4 clouds in our simulation. The different requests can be run on arbitrary clouds. The arrival time of request is differing with each other. Some of the task run as AR mode and rest of them run as best effort mode. We do our simulation locally using these parameters without implementing it in any existing cloud system or using VM interface API.
Our experiment works on two situations loose situation and tight situation; in every case it gives better result than non preemptive algorithm. This pseudo code is better than existing algorithms [3][4] because it gives reservation on resource and task migration over the preemptable resource. The scheduler will re schedule these tasks with predefined probability α. The parameters in table 1.1 are set in simulation randomly to their maximum and minimum values. So we focus on scheduling mechanism.

Table 1.1 RANGES OF PARAMETERS

Parameter	Minimum	Maximum
ETMi,j	27	120
Number of VMs in a cloud	22	120
Number of CPU in a VM	4	10
Memory in a VM	40	2048
Disk space in VM	5000	10000
Speed of copy in disk	100	1000

Result
The figure 3 shows the average response time (a) and average execution time in loose situation, loose situation is where we set arrival time of request far to each other, so there is less resource contention. If any resource contention occur then the best effort job is preempt by AR tasks. In experiment result we find out that it gives better response time and has minimum average execution time.



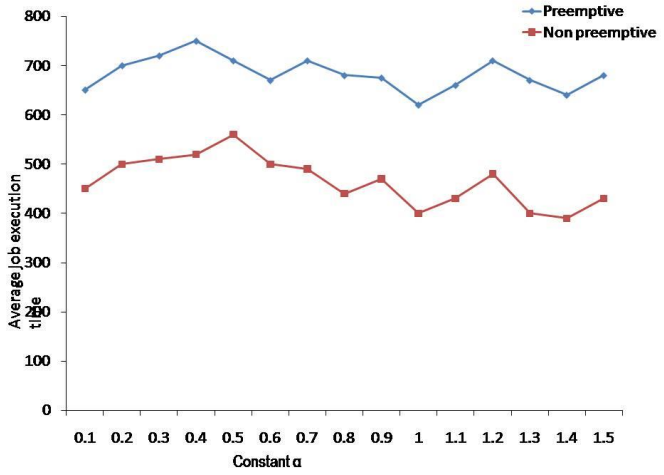
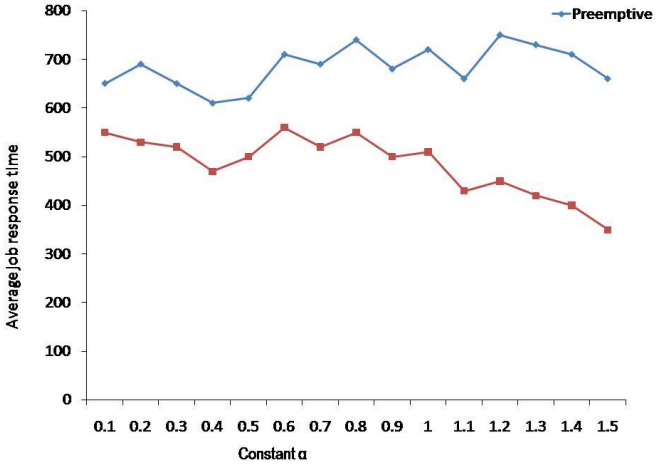


Fig. 3. (a) and (b) average response and execution time in loose situation

The figure 4 shows the tight situation result. In which our given algorithm perform better than existing algorithm [3][4]. For experimental purpose we set arrival time of resource very close to each other. Adaptive procedure works more efficiently in tight situation. The average response time (c) and average execution time (d) is better than previous algorithms.



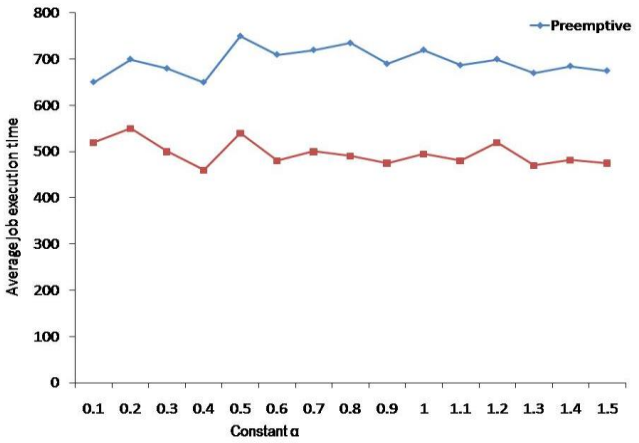


Fig. 4. (c) and (d) average response and execution time in tight situation

The figure 5 shows the experiment result of fault tolerance algorithm, in tight and loose situation of average response time. This algorithm works better in both the situations. When a resource fault is occur during the execution of a process then response time should be very fast by the scheduler. This algorithm tries to assign resources in worst condition, if there are no free and preemptable resources then it gives reservation on resources with a guarantee that it should get resource very soon.

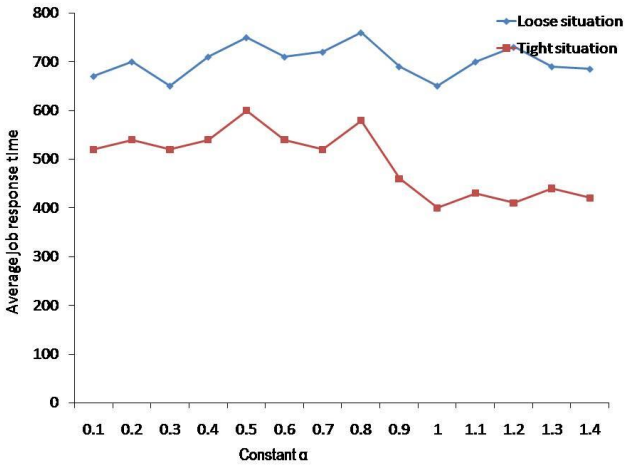


Fig. 5. average response time in loose and tight situation

VI. CONCLUSION

Fault tolerance, QoS, availably, and scalability are still open challenges in cloud computing field. In this paper, we have presented scheduling heuristic for dynamic resource allocation mechanism with resource preemption in cloud. We present a local and global scheduling according to user's service request. We also presents a novel scheme for high priority task it's also beneficial for fault tolerance mechanism in resource management. When a resource gets

failed it immediately provides new resource for task. In this algorithm the priority of task is defined over cost and deadline constraints.

We evaluate our scheduling algorithm using the local simulation. We used evaluation scenarios and test for the resource utilization, deployment of service and fault tolerance.

Future Work- We can works on several parameters which decide the priority of tasks such as min resource requirements, CPU time, Cost and network. We can also investigate energy efficient objective in allocation and utilization of resources. The given heuristic can be also simulating in FOG computing environment. A new scheme can be develop to serve the waiting list. We can also works on mechanism how to assigning slot to tasks.

REFERENCES

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

[2] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in AINA '10: Proceedings of the 2010, *24th IEEE International Conference on Advanced Information Networking and Applications*, pages 400–407, Washington, DC, USA, 2010, IEEE Computer Society.

[3] Vincent C. Emeakaroha, Ivona Brandic, Michael Maurer, Ivan Breskovic, "SLA-Aware Application Deployment and Resource Allocation in Clouds", 35th IEEE Annual Computer Software and Application Conference Workshops, 2011, pp. 298-303.

[4] Chandrashekhar S. Pawar , Rajnikant B. Wagh "Priority Based Dynamic resource allocation in Cloud Computing", 2013 International Conference on Intelligent Systems and Signal Processing (ISSP).

[5] Zhen Xiao, Senior Member, IEEE, Weijia Song, and Qi Chen ," Dynamic Resource Allocation using Virtual Machines for Cloud Computing Environment".

[6] Dorian Minarolli and Bernd Freisleben Department of Mathematics and Computer Science, University of Marburg," Virtual Machine Resource Allocation in Cloud Computing via Multi-Agent Fuzzy Control", 2013 IEEE Third International Conference on Cloud and Green Computing

[7] I. Brandic.Towards self-manageable cloud services. In 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC'09), 2009

[8] R. N. Calheiros, R Ranjan, A.Beloglazov, C.A F.D.Rose, and R. Buyya. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. In Software: Practice and Experience. Wiley Press, New York, USA, 2010.

[9] O.H.Ibarra and C.E.Kim, "Heuristic Algorithms for Scheduling Independent Tasks on Non-identical Processors," Journal of the ACM, pp. 280–289, 1977.

[10]Ravi Jhawar(Graduate Student Member IEEE), Vincenzo Piuri(Fellow, IEEE)and Marco Santambrogio(Senior Member, IEEE), "Fault Tolerance Management in Cloud Computing: A System-Level Perspective" 2013 IEEE International Conference.

[11]V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low level metrics to high level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments," In High Performance Computing and Simulation.

[12]Y. C. Lee, C. Wang, A. Y. Zomaya, and B. B. Zhou.Profit-driven service request scheduling in clouds. In 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010, pages 15 –24, may. 2010.

[13]Shi J.Y., Taifi M., Khreishah A.,"Resource Planning for Parallel Processing in the Cloud," in IEEE 13th International Conference on High Performance and Computing, Nov. 2011, pp. 828-833.

[14]T. Hagras and J. Janecek,"A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems," Parallel Computing, vol. 31, no. 7, pp. 653–670, 2005.

[15]G. Jung, and K. M. Sim, ―Agent-based Adaptive Resource Allocation on the Cloud Computing Environment,‖ in 40th International Conference on Parallel Processing Workshops (ICPPW'11), Taipei City, 2011, pp. 345 – 351.