A Hierarchical Method for Dynamic Resource Allocation in Cloud

T.R.Abinaya
PG Scholar (CSE),
Sri Ramakrishna Engineering College,
Coimbatore,

P.Mathiyalagan,
Assistant Professor(CSE) #2,
Sri Ramakrishna Engineering College,
Coimbatore,

S.N.Sivananda, Ph.D
Emeritus Professor,
Department of CSE,
Karpagam College of Engineering,Coimbator,

ABSTRACT
Cloud Computing has become a buzz word in real world. They provide rapid service to the customer mainly focusing on resource allocation. The main issue of cloud computing is to fix the dynamic resource allocation in order to improve the performance speed and reduce the cost, utilizing the resource efficiently. The main aim of this paper is resource allocation for virtualized cloud environments can improve the performance, availability, guarantees and minimize cost of the energy and time for large cloud service centers. We first formulate from virtual machine allocation for allocation main constrains are load balancing, capacity allocation, frequency scaling, energy efficiency, service differentiation for the efficient resource allocation of virtual machines in the server . Then we move to next class partitioning of incoming large tasks mainly based on weight of the task, budget and resource vector. Based on this method task are efficiently portioned and allocated to virtual machines time of completion of each task is reduced.

Keywords
Hierarchal Framework, Qos-based Load balancing and capacity, Virtual system migration.

1. INTRODUCTION
Cloud computing has become a rapid development mainly for the scalability, flexibility. Task scheduling of incoming task is a major problem in distributed system is based on trusting the trusting accuracy of the incoming resources. In the commercial world they are considering only the cost not about efficient usage of resources. By using resource properly the cost, time and availability of resource are increased gradually. Cloud computing consider both the software (applications) and hardware over the internet. Clouds provide services based on the availability of resources and their demand. They provide services like software, platform and infrastructure areprovided to the consumers based on pay as you go model. Cloud management system can perform all the process like resource allocation and also application deployment in cloud.

In previous days providing services through the centralized based service but by the hierarchical method can solve problem in any step and NP-hard problem also overcome waiting for allocation of each task has been overcome. In [10][2][11] they have consider mainly reducing the cost, improve the performance, and also availability and scalability and dependability. By have consider first based on incoming task partitioning and allocating by considering load balancing, capacity allocation, frequency scaling, energy efficiency, service differentiation

The concept of virtualization is based on sharing the same physical resource by the multiple end user this is the fundamental police to developed to manage the cloud systems. By using virtual machines cost of physical machines can be reduced and also higher utilization of resources.

The remaining part of the paper is organized as follows: Section 2 previous work from the literature, while Section 3 introduces our hierarchical framework. The optimization formulations and their solutions in Sections 4 and 5, respectively. Section 6 experimental results, with concluding remarks in Section 7.

2. RELATED WORK
Bernardetta Addis, DaniloArdagna, Barbara Panicucci, Mark S. Squillante, Fellow, IEEE, and Li Zhathey [1] “A Hierarchical Approach for the Resource Management of Very Large Cloud Platforms”, in this paper they have proposed based on considering only the weight of the incoming task to portioning the resource because of this they may have various conflicts and for allocation of resource to virtual machines they have only consider the capacity allocation, frequency, scaling because of this efficiency of resource allocation has been reduced.

D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang, “Energy-Aware Autonomic Resource Allocation in Multitier Virtualized Environments”, in this paper they proposed based on prediction of incoming task.They mainly focused on various timescales [16] they have compare tasks based on transaction between frequency and overhead of these decision .Self-managing frameworks [5], [16] they decide to themselves whether virtual machines from the server can shut down or power up a server, and also VM migrations from one to another for these actions they required significant energy so they won’t performed so frequently.

T. Nowicki, M.S. Squillante, and C.W. Wu, “Fundamentals of Dynamic Decentralized Optimization in Autonomic Computing Systems”, in this paper, they have a perspective of a PaaS provider manages complete the transaction service of the customer task to satisfy response time and availability and to reduce energy costs in very large cloud service centers. They propose a distributed hierarchical framework [17] based on a mixed-integer nonlinear optimization of resource management across multiple timescales. At the top hierarchy, a central manager (CM) partitions of the incoming workloads based on some constrains and resources (physical servers) over a 24-hour horizon to create clusters with homogeneous workload profiles one day in advance. So in each cluster, an application manager (AM) allocates the resources in centralized manner in hourly basis to the subsystems. AMs have to decide the which application have to executed by each server of the cluster placing application to the subsystems they check for the requested volume of resources and availability of the resource in the cluster have to check the capacity required to the execution application at each server. AMs can also decide whether server can switch to a low-power sleep state based on the cluster load or to reduce the operational frequency of servers.

A recent study [12] they proposed hierarchical control solutions, pin this they mainly focused on cluster-level control architecture that perform multiple server power controllers within a virtualized server cluster [12]. The higher layer determines capacity allocation of VM in server and also VM migration within a cluster, while the inner controllers have to check the power level of individual servers. However, application and VM placement within each service center cluster is assumed to be given before execution. For the completion of the task is mainly based on the coordination among multiple power controllers acting at rack enclosures and at the server level is done.

Energy-Aware Autonomic Resource Allocation in Multitier Virtualized Environments [16] in this have been proposed to achieve fine-grained dynamic resource provisioning method. They proposed scheduling the tasks based on Xen's credit scheduler. They can dynamically control the usage of resource like disk I/O bandwidth among the allocated virtual machines. Through these methods for VMs can allocate resources are used for dynamic partition or else reassembled of tasks to meet the needs of the users. These are the unique future of the VMs in cloud which is not possible to apply in most Grid systems [6], [7], [8].

A Dynamic Optimization Algorithm for Task Scheduling in Cloud Computing With Resource Utilization [18] Ram Kumar Sharma, Nagesh Sharma they proposed mainly on scheduling tasks and efficently allocating the resources to the virtual machines The objective of this paper is to the maximum utilization on client and server side accessing the cloud environment

3. HIERARCHICAL APPROACH FOR THE RESOURCE MANAGEMENT

In this method by allocating the resource in apex to base manner, when request is coming by allocated the resource for the incoming task is going to partitioning based on the weight of the task, budget, resource vector and given to suitable VMs based on each cluster in the application manager they compare with load balancing, capacity allocation, frequency scaling, energy efficiency, service differentiation for the efficient resource allocation of virtual machines in the server.

The PaaS provider have to support the multiple transactional services to execute , each transaction with different customer with their different application. The hosted services can be heterogeneous with respect to resource demands, workload profile, network bandwidth and QoS requirements. Services with different property of workload profiles are separated into independent request classes, where they set serves a set R of request classes. Fig. 1 architecture with hierarchical approach. The system includes a set S of heterogeneous servers, each running a virtual machine monitor (VMM) (such as VM Ware or Xen, IBM POWER Hypervisor), the computing resources are capped and reserved for the execution of individual VMs. The physical resources like CPU, hard disks, bandwidth of a server are partitioned among each based on weight of the task, budget, resource vector to create clusters of virtual machines.

Resource Allocation:

• Server ON/OFF

• Applications placement

• Frequency Scaling (DVFS)

• Load balancing

• Capacity allocation

So now any class can support any type of application in the multiple tiers. Each VM is hosted in a single application tier, where also multiple VMs are in the same application tier can be run in parallel on different hosts. Under very less work load conditions server move to standby mode or to shut down mainly to reduce the energy cost. These servers move back to active state during the peak hours.

Accept that each server has a single CPU supporting dynamic voltage and frequency scaling (DVFS) by choosing both its supply voltage and operating frequency from a limited set of values, noting that this single-CPU supports that is without loss of overview in heavy traffic and can be easily relaxed. By using DVFS is not became overhead to system but for hibernating and restoring a server both require time and energy. In this [13], they proposed adopt full system power models and assume that the power consumption of a server depends on its operating frequency/voltage as well as the current CPU utilization. Our resource management framework is based on a hierarchical architecture [17]. At the highest level of the hierarchy, a CM acts on (i.e., every 24 hours) and is responsible for partitioning the classes and servers into clusters (long-term problem). The object is to form clusters C, based on same work load profile it reduces the server switching finer-grained timescales. Furthermore, by denote by R_c the set of request classes assigned to cluster C and by S_c the set of physical servers in cluster C at time interval t. At a lower level of the hierarchy, AMs centrally manage individual clusters (i.e., on an hourly basis). Each AM can decide (medium-term problem):

1. Placing the applications within the cluster,

2. Managing the VMs capacities for running on each server.

3. Load balancing supports in same application tier for various VMs.

4. Switching servers into active or low-power sleep states, and

5. Increasing/decreasing the operating frequency operation of the CPU of a server.

Where

C	Set of clusters
A_r	Set of applications involved in the execution of class r
R	Set of requested classes
S	Set of servers available at the cloud service center
F_s	Set of operating frequencies for server s
$C_{s,f}$	Capacity of server s when it is working at frequency f
U_s	Server s CPU utilization
U_r	Per requested revenue
RAM_S	RAM available at server s
$RAM_{\tau,r}$	RAM required for the execution of the application tier τfor class r
u_r	Requested class r maximum utility function value
α_r	Requested class r utility function slope
$\bar{A}_r$	Class r availability threshold
$\bar{R}_r$	Class r revenue region threshold
Cm	Cost associated with moving VMs to a different server
a_s	Server s available
$\bar{c}_s$	Time unit cost for server s when it is in low power sleep state
CS_s	Cost for switching server s from low power sleep to active state
$\mu_{\tau,r}$	Maximum services rate of a capacity 1 server for executing application at tier τ for class r request
$t_{i,j}$	j task submitted to p_i
$r^*_{(j)k}$	Optimal vector with availability constrain $t_{i,j}$
$r_{(j)k}$	Resource vector allocated to $t_{i,j}$

Both dependability and fault tolerance are important issues in developing resource management for very large infrastructures. There has been research employing peer-to-peer designs [10] or using gossip protocols [12] to increase dependability. Wuhib et al. [12] propose a gossip protocol to compute, in a distributed and continuous fashion, a heuristic solution to a resource allocation problem for a dynamically changing resource demand. In our solution, to provide better dependability for the management infrastructure, each cluster maintains a primary and a backup AM. The primary and the backup do not need to be tightly synchronized to the second level, as the decisions they are making are at a timescale of 5-15 minutes. Similarly, a backup CM is also maintained.

Perspective of a Platform as a Service provider which has to manage the applications of its end customers:

• Providing response time and availability guarantees

• Minimizing its energy costs

• Propose a distributed hierarchical solution based on mixed integer non-linear optimization for the management of resources of very large cloud service centers, acting at multiple time-scales

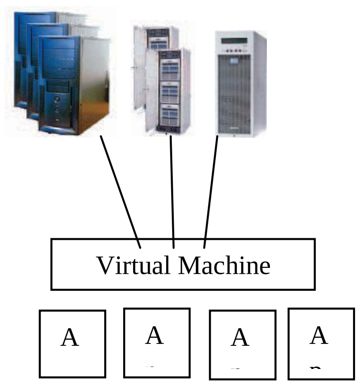

Fig1: System Architecture

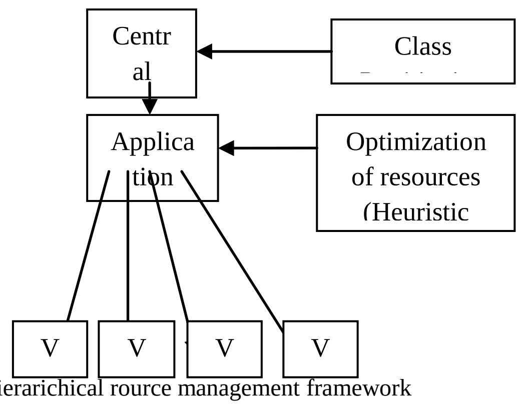

Fig2:Hierarichical rource management framework

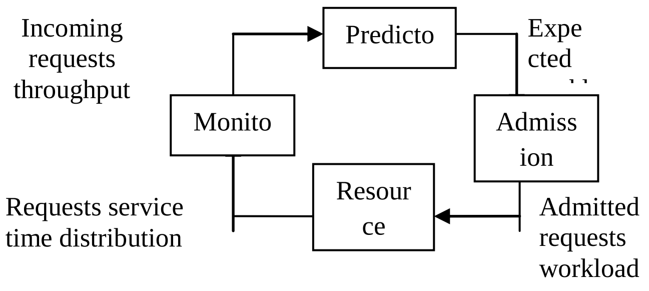
Incoming requests throughput
Predicto
Expe cted
Monito
Admiss ion
Resour ce
Requests service time distribution
Admitted requests workload

Fig3:Resource Allocation

4. FORMULATION OF OPTIMIZATION PROBLEMS

In this section, [1]focus on resource management optimization problem the main objective is to maximize profit, namely the difference between revenues from SLA contracts and costs associated with servers switching and VM migrations, while providing availability guarantees.

In Resource Allocation Optimization Algorithm works in Application Manager where they decreses the work time and perform the work faster by using our algorithm.

4.1 Resource Allocation () optimization procedure

```
(x,y,z,φ,λ)=Initial_Soultion();
FPI(φ,λ);
STOP=false;
While not STOP do
While improve do
While improve do
```

```
Local_Search_withAvailabilty(x,y,z,φ,λ);
End
FPI(φ,λ);
End
        $(\hat{x},\hat{y},\hat{z},\hat{\varphi},\hat{\lambda})$=Save_Best_Solution();
While iteration_limit do
Local_Search_withAvailabilty(x,y,z,φ,λ);
End
Remove_Violations(x,y,z,φ,λ);
If not_feasible_solution then
STOP=true;
(x,y,z,φ,λ)= $(\hat{x},\hat{y},\hat{z},\hat{\varphi},\hat{\lambda})$;
End
End
return(x,y,z,φ,λ);
```

Three basic components comprise our overall solution approach:

Step 1. An initial solution is performed by applying a greedy algorithm whose output is the set of active servers x, their corresponding operating frequencies y, the assignment of tiers to servers z, an initial capacity allocation and an initial load balancing vector.

Step 2. A fixed-point iteration (FPI) is used to search for improvements on the initial capacity allocation and load balancing decisions. FPI solves the load balancing and capacity allocation problems.

Step 3. A local search procedure iteratively improves the latest solution by exploring its neighborhood and using so-called moves.

By which they are improving the allocation of resources based on their requirement and they complete the task within the time.

Intial Solution

The process is based on greedy approach iteratively find best solution to find the capacityand load balancing of the server.where they find the appication tier based on $W_{r,r} = \Lambda_{r,r}^{res} \sum_{r \in Ar} \frac{1}{\mu_{r,r}}$ where $\Lambda_{r,r}^{res}$ allocating the workload to be allocated and also order the server based on no decreasing order

Fixed Point Iteration

In this they are finding the best solution for both the problem and set the new value for the load balancing and capacity allocation. Where they update the value based on local search neighborhood().

Local Search Implementation

By using local search method find the best solution for the φ and λ if the best value is not obtained then again perform the process if no more improvement in the process then stop the local search

4.2 Class partitioning Algorithm

```
Input: R set of classes
Output: P = {$P_1$…….$P_N$} set of partitions
$L_1$ =0, $L_2$=0;
For r← 1 to |R| do
        $L_1$←{r};
End
Foreach couple ($P_i$,$P_j$), $P_i$€ $L_1$, $P_j$€ $L_1$ do
Evaluate AR($P_i$,$P_j$); $L_2$← $P_i$U$P_j$ (AR order);
End
Repeat
Improve =false ;
Foreach couple ($P_i$,$P_j$), € $L_2$ do
If AR($P_i$,$P_j$)>τ and | $P_i$|+| $P_j$ |<M then
$\bar{P}$= $P_i$U$P_j$;
Foreach P€ $L_2$ s.t. $P_i$ € P or $P_j$€ P do
$L_2$= $L_2$\P;
End
Foreach couple ($\bar{P}$,$P_i$)
Evaluate AR($\bar{P}$, Pi), $L_2$←$\bar{P}$ U$P_i$ (AR order);
End
$L_1$←$\bar{P}$;improve =true;
End
End
Until Improve;
```

Long-Term Solution

Our hierarchical optimization framework is composed of a CM and a set of AMs . The AM controls a subsystem of the cloud service center, operating on a cluster that includes a subset of applications and resources (physical servers). Each AM optimally assigns the available servers within its cluster to the given subset of applications, employing the algorithm described in the previous section.

4.3 SLICE HANDLER (PSM)

```
For (k=1 → R) do
Sum_allocation=$\sum_{p=1}^{M} r *_{(p)k}^{i}$;
For(each $t_{(j)}$.j=1,2,……..M) do
$r_{(j)k}^{**} = r_{(j)k}^{*} r_{(j)k}^{*}$/sum_allocation).$C_k$;
Assign $r_{(j)k}^{**}$ to $t_{(j)}$;
End for
End for
Notify VMM to readjust resource allocation
among all running tasks;
```

4.4 EVENT HANDLER

```
If(The event is the arrival of a scheduled task $t_{(x)}$ ) then
$a_{(x)}$ = c($p_s$)-$\sum_{j=1}^{x-1} r_{(j)}^{*}$;
conduct Algorithm 1 for $t_{(x)}$;
```

```
end if
if (The event is the completion of a  task $t_{(y)}$) then
a($p_s$) =a($p_s$) + $r^*_{(y)}$;
for(each $t_{(i)}$ still running on $p_s$) do
if(such that k $r^*_{(j)k}$<$r^{(*)}_{(j)k}$,k=1…R) then
conduct Algorithm 1 for $t_{(i)}$;
end if
end for
end if
```

Table1:Resource Utlization Ratio

|R||S|	Class Partitioning %	Slice and Event handler %
20,20	11	0.24
40,400	11	0.49
80,800	10.81	4.45
100 ,1000	9.98	6.97
120 ,1200	7.75	7.62

In this algothim partitioning our algothim based workload profile only but in this algothim based on weight of the task , budget, resource vector.

The two main [11] procedures: 1) Slice handler: It is activated periodically to equally scale the amount of resources allocated to tasks, such that each running task can acquire additional resources proportional to their demand along each resource dimension.

2) Event handler:It is responsible for resource redistribution upon the eventsof task arrival and completion. The pseudocodes are shownin the algorithms The slice handler is periodically performed by ps's VMM, while the event handler is only invoked upon task arrival or completion.

5.EXPERIMENTAL RESULTS

By comparing with class partitioning with slice and event handler based on incoming work partitioning the work load. Here by compaing with various intances with the same size of the task. Results shows that slice method perform partitioning faster than the existing class partitioning method.

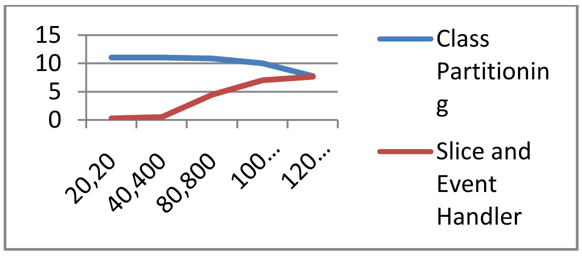

Fig4: Resource Utlization Ratio

The CPU utlization of the resources with the thersold value between 40 to 60 and with the proper utilization of the present resource.

6. CONCLUSION

Optimization of task's resource allocation under user's budget: With a realistic monetary model, propose a solution which can optimize the task execution performance based on its assigned resources under the user budget. prove its optimality using the KKT conditions. Maximized resource utilization based on Slice and Event Handler. In order to further make use of the idle resources, we design a dynamic algorithm by combining the above algorithm with PSM and the arrival/completion of new tasks. This can give incentives to users by gaining an extra share of unused resource without more payment. Experiments confirm achieving a super optimal execution efficiency of their tasks is possible. Very large scales systems have also been analyzed. In current work, are extending the resource allocation problem by considering, at a finer-grained timescale, the adoption of pure control theory models at timescales of seconds. So by using slice and event handler method we are allocating the resources efficiently to the virtual machines than our class partitioning method. Because of that time and cost is reduced automatically.

7. REFERENCES

[1] Bernardetta Addis, DaniloArdagna, Barbara Panicucci, Mark S. Squillante, fellow, ieee, and li zhang ,"A Hierarchical Approach for the Resource Management of Very Large Cloud Platforms", ieee transactions on dependable and secure computing, vol. 10, no. 5, september/october 2013

[2] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, I.S.A. Rabkin, and M. Zaharia,"Above the Clouds: A Berkeley View of Cloud Computing," http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS- 2009-28.pdf, 2013.

[3] M.D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," IEEE Internet Computing, vol. 13, no. 5, pp. 10- 13, Sept./Oct. 2009.

[4] Gartner, "2012 Cloud Computing Planning Guide," http:// my.gartner.com/portal/server.pt?open=512&objID=249&mode=2&PageID=864059&resId=1837017&ref=Browse, 2013.

[5] D. Kusic, J.O. Kephart, N. Kandasamy, and G. Jiang, "Power and Performance Management of Virtualized Computing Environments via Lookahead Control," Proc. Int'l Conf. Autonomic Computing (ICAC), 2008.

[6] F. Longo, R. Ghosh, V.K. Naik, and K.S. Trivedi, "A Scalable Availability Model for Infrastructure-as-a-Service Cloud," Proc. IEEE/IFIP 41st Int'l Conf. Dependable Systems and Networks (DSN), 2011.

[7] G.S.G.E.D. Lazowska, J. Zahorjan, and K.C. Sevcik, Quantitative System Performance, Computer System Analysis Using Queueing Network Models. Prentice-Hall, 1984.

[8] H. Li and S. Venugopal, "Using Reinforcement Learning for Controlling an Elastic Web Application Hosting

Platform," Proc. ACM Eighth Int'l Conf. Autonomic Computing (ICAC), 2011.

[9] Microsoft, "Windows Azure," http://msdn.microsoft.com/ en us/library/windowsazure/dd163896, 2013.

[10] C. Adam and R. Stadler, "Service Middleware for Self-Managing Large-Scale Systems," IEEE Trans. Network and Service Management, vol. 4, no. 3, pp. 50-64, Dec. 2007.

[11] J. Cao, K. Hwang, K. Li, and A.Y. Zomaya, "Optimal Multiserver Configuration for Profit Maximization in Cloud Computing," IEEE Trans. Parallel Distributed Systems, preprint, no. 99, 2012.

[12] F. Wuhib, R. Stadler, and M. Spreitzer, "A Gossip Protocol for Dynamic Resource Management in Large Cloud Environments," IEEE Trans. Network and Service Management, vol. 9, no. 2, pp. 213- 225, June 2012.

[13] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No 'Power' Struggles: Coordinated Multi Level Power Management for the Data Center," SIGARCH Computer Architecture News, vol. 36, no. 1, pp. 48-59, 2008.

[14] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A Comparison of High-Level Full-System Power Models," Proc. Conf. Power Aware Computing and Systems (HotPower), 2008.

[15] Sheng Di, Cho-Li Wang," Dynamic Optimization OfMultiattribute Resource Allocation In Self-Organizing Clouds" IEEE Transactions On Parallel And Distributed Systems.

[16] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang, "," IEEE Trans. Services Computing, vol. 5, no. 1, pp. 2-19, Jan.-Mar. 2012.

[17] T. Nowicki, M.S. Squillante, and C.W. Wu, "Fundamentals of Dynamic Decentralized Optimization in Autonomic Computing Systems," Self-Star Properties in Complex Information Systems, pp. 204-218, Springer-Verlag, 2005.

[18] Saure D, Sheopuri A, Qu H, Jamjoom H, Zeevi A (2010)," Time-of-use pricing policies for offering cloud computing as service," in IEEE SOLI 2010, pp 300–305.