International Journal of Technical Innovation in Morden Engineering & Science (IJTIMES)
Impact Factor: 3.45 (SJIF-2015), e-ISSN: 2455-2584
Volume 3, Issue 6, June-2017

# Better QoS using Resource Provisioning Augmentation in Cloud Computing

Hitesh Bheda[1], Prakruti Thakkar[2]

[1]*Indus University*
[2]*Gujarat University*

***Abstract— Cloud Computing is a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized Service Level Agreements. Resource Provisioning is an important and challenging problem of Cloud Computing. It plays an important role in the performance of the entire system and also the level of customer satisfaction provided by the system. There are many resource provisioning techniques, both static and dynamic each one having its own advantages and also some challenges. These resource provisioning techniques used must meet Quality of Service (QoS) parameters like availability, throughput, response time, security, reliability etc., and thereby avoiding Service Level Agreement (SLA) violation. In this paper, survey on Static and Dynamic Resource Provisioning Techniques is made. Here we have taken Round robin algorithm as a resource allocation algorithm for dynamic resource provisioning.***

***Keywords— Cloud Computing, Resource Provisioning, Virtual Machines, Aneka, Round Robin Algorithm.***

## I. INTRODUCTION

Cloud computing [1] led to an innovative approach in the way in which IT infrastructures, applications, and services are designed, developed, and delivered. Three major services offerings contribute to define Cloud computing: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Infrastructure-as-a-Service providers deliver on-demand components for building IT infrastructure such as storage, bandwidth, and most commonly virtual servers, which can be further customized with the required software stack for hosting applications. Platform-as-a-Service providers deliver development and runtime environments for applications that are hosted on the Cloud. They allow abstraction of the physical aspects of a distributed system by providing a scalable middleware for management of application execution and dynamic resource provisioning. Software-as-a-Service providers offer applications and services on-demand, which are accessible through the Web. SaaS applications are multi-tenant and are composed by the integration of different components available over the Internet.

In cloud computing, an effective resource allocation strategy is required for achieving user satisfaction and maximizing the profit for cloud service providers. Multiple cloud users can request number of cloud services simultaneously in cloud computing. Provision must be provided such that all resources are made available to user's request in efficient manner to satisfy their needs. This resource provisioning can be done by using virtual machines that handles user's request and provide appropriate services to the requested users [2].

The rest of the paper is organized as follows: Section II discusses Resource provisioning techniques. Section III discusses about the cloud computing framework Aneka and Section IV discusses about the resource provisioning through load balancing algorithm.

## II. RESOURCE PROVISIONING IN CLOUD COMPUTING

Resource provisioning means the selection, deployment, and run-time management of software (e.g., database management servers, load balancers) and hardware resources (e.g., CPU, storage, and network) for ensuring guaranteed performance for applications. This resource provisioning takes Service Level Agreement (SLA) into consideration for providing service to the cloud users. This is an initial agreement between the cloud users and cloud service providers which ensure Quality of Service (QoS) parameters like performance, availability, reliability, response time etc. Based on the application needs Static Provisioning/Dynamic Provisioning and Static/Dynamic Allocation of resources have to be made in order to efficiently make use of the resources without violating SLA and meeting these QoS parameters. Over provisioning and under provisioning of resources must be avoided. Another important constraint is power consumption. Care should be taken to reduce power consumption, power dissipation and also on VM placement. There should be techniques to avoid excess power consumption.

*1. Static Provisioning*

For applications that have predictable and generally unchanging demands/workloads, it is possible to use "static provisioning" effectively. With advance provisioning, the customer contracts with the provider for services and the provider prepares the appropriate resources in advance of start of service. The customer is charged a flat fee or is billed on a monthly basis.

*2. Dynamic Provisioning*

In cases where demand by applications may change or vary, "dynamic provisioning" techniques have been suggested whereby VMs may be migrated on-the-fly to new compute nodes within the cloud. With dynamic provisioning, the provider allocates more resources as they are needed and removes them when they are not. The customer is billed on a pay-per-use basis. When dynamic provisioning is used to create a hybrid cloud, it is sometimes referred to as cloud bursting.

*3. User Self-Provisioning*

With user self-provisioning (also known as cloud self-service), the customer purchases resources from the cloud provider through a web form, creating a customer account and paying for resources with a credit card. The provider's resources are available for customer use within hours, if not minutes.

## III. CLOUD ENVIRONMENT: ANEKA

Aneka is a platform and a framework for developing distributed applications on the Cloud. It can manage the CPU cycle in desktops, servers, data centers on demand. It has many APIs and programming abstractions for transparently exploiting such resources. It has tools for monitoring and controlling the deployed infrastructure. It can be used as a public cloud or private cloud. Fig. 1 represents the architecture of the Aneka:

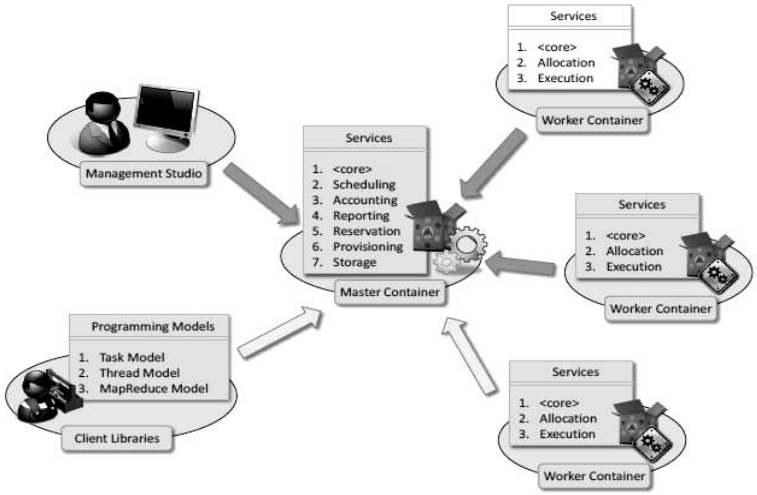


*Fig. 1 A sample line graph using colors which contrast well both on screen and on a black-and-white hardcopy*

Aneka is a Platform-as-a-Service (PaaS) that facilitates the development, deployment and monitoring of distributed applications in clouds. For this purpose, Aneka has different components that are described below [3]:

- *Aneka Client Libraries:* These are Application Programming Interfaces (APIs) created to build applications based on different distributed programming models including Task, Thread and Map-Reduce models. After developing an application using these APIs, it can be easily deployed and executed in Aneka managed clouds.
- *Aneka Cloud:* This is the virtual network that is composed by the Aneka Master and the Aneka Worker Containers which work together to run the applications. The resources used for the Aneka cloud can be provisioned from different sources such as private and public clouds, networks of computers, or multicore servers.
- *Aneka Containers:* These are the Aneka components that are deployed on different machines that are part of the Aneka cloud. These containers host and offer different services that vary according to the two roles that can be installed: Aneka Master and Aneka Worker. An important feature of Aneka containers is that they are developed using a Service Oriented Architecture (SOA), so containers are composed of a core and a number of services that can be easily plugged and unplugged according to the needs of the user applications. Furthermore, it is possible to create new services or replace the existing ones. This makes Aneka a highly customizable and extensible system.
- *Aneka Master:* This is an Aneka container whose main responsibility is orchestrating and monitoring the execution of the application created by the user. Therefore, its default services are scheduling, provisioning, storage, reporting and accounting.
- *Aneka Worker***:** This is an Aneka container that is responsible for executing every individual task that constitutes the user application. The default services associated to this role are allocation and execution.
- *Aneka Management Studio:* This is a graphical interface that allows the construction and monitoring of the Aneka cloud. Static and dynamic Aneka clouds can be configured. In the static cloud, the required resources are added manually while in the dynamic cloud an Aneka Master is set up with a provisioning service to request and release machines dynamically from cloud providers.

The monitoring capabilities of Aneka are supported via the reporting service, the container core and the Aneka Management Studio. The Aneka Management Studio acts as the central point that communicates with the containers and its services to generate reports to observe the status of the containers, resource utilization details and associated cost of the resources used.

*Resource Provisioning Implementation with Aneka:*

The main functionality of the provisioning service is to provision and release resources that are acquired from different cloud providers. To satisfy this requirement, Aneka offers abstract interfaces that can be used to implement the logic to invoke cloud provider interfaces to provision and release resources.

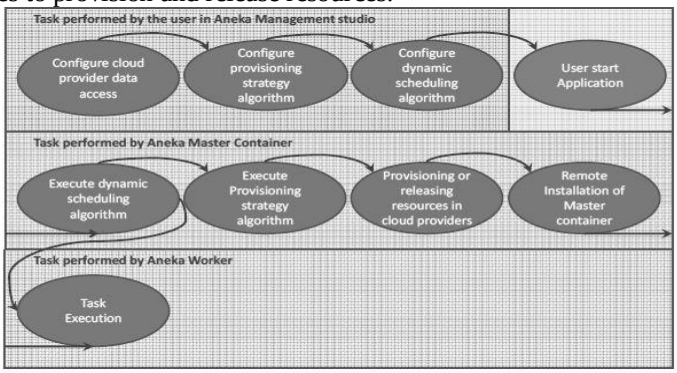


*Fig. 2 Complete Resource Provisioning in Aneka*

Additionally, Aneka provides standard editors to configure the access data to the cloud providers and the virtual machines features. These configurations are used to create a pool of connections that can be invoked any time such that resources from a specific cloud provider can be acquired or released.

In addition to the pool of connections to cloud providers these service also needs to be configured with an algorithm that specify the strategy to select the cloud provider that should be used to acquire the next resource to be provisioned. This strategy is invoked by scheduling algorithms when the resources are acquired dynamically by the Aneka Master Container. When the provisioning service is invoked by the Aneka Management Studio, the resources are added manually and statically by the user. In this case, the user specifies the cloud provider that he wants to use to provision the virtual machine and the provisioning strategy algorithm is not necessary.

We have installed the xenserver on a machine as a bare-metal hypervisor. The xenserver is accessed using the xencenter, which can be installed on any other system with windows OS and it is connected through LAN to the xenserver.

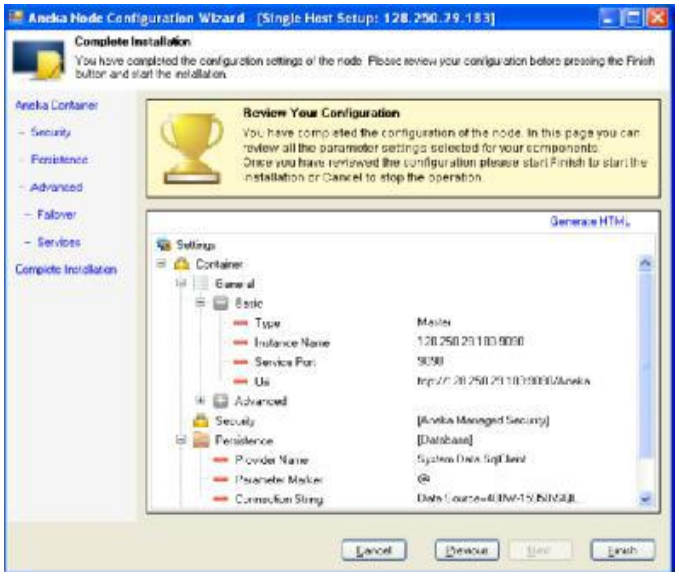


*Fig. 3 Aneka Node Configuring Wizard*

Now we have to create the virtual machines (VMs) in the server using the xencenter. The VM can be installed either using the DVD drive of server, or by a shared library in the network which has the ISO files. The storage is created by putting all the needed ISO files in a place and the folder is shared to everyone on the network, by that anyone on the network can access that folder using its IP address. In xencenter by using the option called add storage we can add ISO location to your xencenter. While creating the VMs give this storage as the source for the OS needed by the virtual machines. After installing the VMs next step is installing Aneka in VMs. We have to install xenserver tools provided by the xenserver for observing the CPU and memory utilizations of the Virtual Machine created.

Aneka setup starts by creating a machine with some valid credentials. We have a default local repository. In that machine create a daemon and install a container which shows in the context menu. While installing aneka choose the option either master or slave. For one VM choose as a master and choose all others as the slaves. We just need to have a connection to the server at client side. Write a program using aneka with SDK provided by it and compile it. There is no need to install the aneka software at client side. Here he gains the software as a service. And the platform client is using may be anything, the server side we created two kinds of platforms (we can have many). Here the client gets the infrastructure as a service. The client connects with aneka master node at server side, by giving the credentials (we can

give pricing details while installing the aneka in server). The process runs on server by using slaves if needed. And the total resource managing is done by the master node.

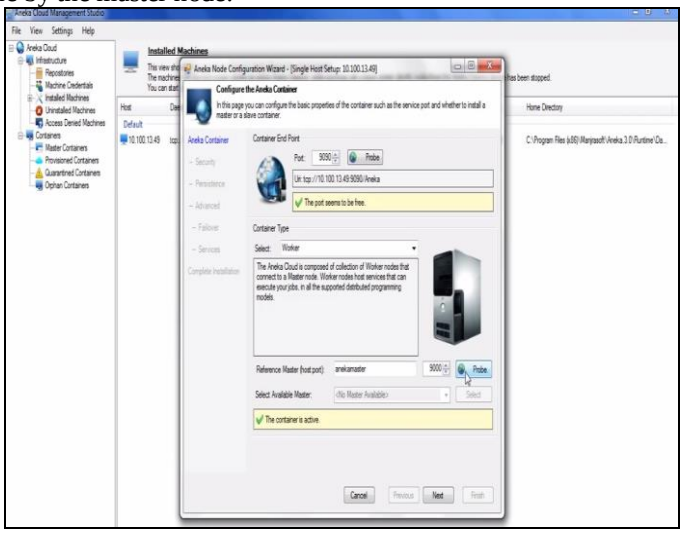

*Fig. 4 Aneka master node creation*

The Round robin algorithm is suitable for web applications where response time is one of the important factors. For web applications guaranteeing average response time is difficult because traffic patterns are highly dynamic and difficult to predict accurately and also due to the complex nature of the multi-tier web applications it is difficult to identify bottlenecks and resolving them automatically. This provisioning technique proposes a working prototype system for automatic detection and resolution of bottlenecks in a multi-tier cloud hosted web applications. This improves response time and also identifies over provisioned resources. VM based resource management is a heavy weight task. So this is less flexible and less resource efficient. To overcome this, a lightweight approach called Elastic Application Container [EAC] is used for provisioning the resources where EAC is a virtual resource unit for providing better resource efficiency and more scalable applications. This EAC–oriented platform and algorithm is to support multi tenant cloud use.

Dynamic creation of the tenant is done by integrating cloud based services on the fly. But dynamic creation is by building the required components from the scratch. Even though multitenant systems save cost, but incur huge reconfiguration costs. This approach allows clients to specify their requirements which are not supported in previous techniques. This approach proposes a novel user interface-tenant selector (UTC) model which enables cloud based services to be systematically modeled and provisioned as variants of existing service tenants in the cloud. This considers functional, non functional and resource allocation requirements which are explicitly specified by the client via the user interface component of the model. So the cost and time is saved in this approach. The technique proposed makes use of the provisioner called adaptive power-aware virtual machine provisioner (APA-VMP) where the resources are provisioned dynamically from the resource pool.

This is from Infrastructure-as-a- Service provider point of view where the custom Virtual machines (VM) are launched in appropriate server in a data center. The cloud data center considered is heterogeneous and large scale in nature. The proposed meta scheduler maps efficiently a set of VM instances onto a set of servers from a highly dynamic resource pool by fulfilling resource requirements of maximum number of workloads.

This technique reduces power consumption without affecting performance. Server Consolidation is a technique to save on energy costs in virtualized data centers. The instantiation of a given set of Virtual Machines (VMs) to Physical Machines (PMs) can be thought of as a provisioning step where amount of resources to be allocated to a VM is determined and a placement step which decides which VMs can be placed together on physical machines thereby allocating VMs to PMs. Here a provisioning scheme is proposed which takes into account acceptable intensity of violation of provisioned resources.

Correlation among aggregated resource demands of VMs is considered when VMs are mapped to PMs. This reduces number of servers (up to 32%) required to host 1000 VMs and thus enables to turn off unnecessary servers. In Cloud Computing federated Cloud Environment is used when the resource requirement of user requests exceeds the resource limits of Cloud Providers' resources. It is desirable to reduce SLA violation which can be achieved through load balancing algorithm that is threshold based. This algorithm allocates VMs in order to balance the load among multiple datacenters in a federated cloud environment by focusing on reducing users' SLA violation.

Brief reviews of few existing resource allocation algorithms are presented in the following:

*Token Routing* [8]: The main objective of the algorithm is to minimize the system cost by moving the tokens around the system. But in a scalable cloud system agents cannot have the enough information of distributing the work load due to communication bottleneck. So the workload distribution among the agents is not fixed. The drawback of the token routing algorithm can be removed with the help of heuristic approach of token based load balancing. This algorithm provides the fast and efficient routing decision. In this algorithm agent does not need to have an idea of the complete knowledge of their global state and neighbor's working load. To make their decision where to pass the token they actually build their own knowledge base. This knowledge base is actually derived from the previously received tokens. So in this approach no communication overhead is generated.

*Round Robin:* In this algorithm, the tasks are divided between all VMs. Each task is assigned to the VM in a round robin order. The task allocation order is maintained locally independent of the allocations from remote processors. Though the work load distributions between processors are equal but the job processing time for different processes are not same. So at any point of time some nodes may be heavily loaded and others remain idle. This algorithm is mostly used in web servers where Http requests are of similar nature and distributed equally.

*Randomize* [9]: Randomized algorithm is of type static in nature. In this algorithm a process can be handled by a particular node n with a probability p. The process allocation order is maintained for each processor independent of allocation from remote processor. This algorithm works well in case of processes are of equal loaded. However, problem arises when loads are of different computational complexities. Randomized algorithm does not maintain deterministic approach. It works well when Round Robin algorithm generates overhead for process queue.

*Central queuing* [9]: This algorithm works on the principal of dynamic distribution. Each new activity arriving at the queue manager is inserted into the queue. When request for an activity is received by the queue manager it removes the first activity from the queue and sends it to the requester. If no ready activity is present in the queue the request is buffered, until a new activity is available. But in case new activity comes to the queue while there are unanswered requests in the queue the first such request is removed from the queue and new activity is assigned to it. When a processor load falls under the threshold then the local load manager sends a request for the new activity to the central load manager.

*Connection mechanism* [9]: Load balancing algorithm can also be based on least connection mechanism which is a part of dynamic scheduling algorithm. It needs to count the number of connections for each server dynamically to estimate the load. The load balancer records the connection number of each server. The number of connection increases when a new connection is dispatched to it, and decreases the number when connection finishes or timeout happens.

Round Robin Algorithm is by far the simplest algorithm available to distribute load among nodes. It is therefore often the first choice when implementing a simple scheduler. One of the reasons for it being so simple is that the only information needed is a list of nodes [7]. However this is only when several key assumptions are true:

1. The nodes must be identical in capacity. Otherwise performance will degrade to the speed of slowest node in the cluster.
2. Two or more client connections must not start at the same time. Should they, the node chosen will be the same, because the order of nodes retrieved from the cluster is the same every time.
3. The jobs must be similar to achieve optimum load distribution among the nodes. If a single node is more loaded than others it will become a bottleneck in the system [7].

## IV. RESULT ANALYSIS

The Round Robin Algorithm is implemented for simulation. Table 1 shows the result based on Round Robin VM Load Balancing algorithm for overall response time of the cloud. In this min (ms) time, max (ms) time to different number of virtual machines are analyzed. Table 4 shows the result based on Round Robin VM Load Balancing algorithm for Data Centre processing time of the cloud. In this min (ms) time, max (ms) time to different number of virtual machines are analyzed.

TABLE I
OVERALL RESPONSE TIME FOR ROUND ROBIN

| No of VMs | Avg(ms) | Min(ms) | Max(ms) |
|---|---|---|---|
| 10 | 301.45 | 238.06 | 370.12 |
| 20 | 301.12 | 238.06 | 370.12 |
| 30 | 301.34 | 238.14 | 370.12 |
| 40 | 301.53 | 238.13 | 371.02 |
| 50 | 301.61 | 238.14 | 371.02 |

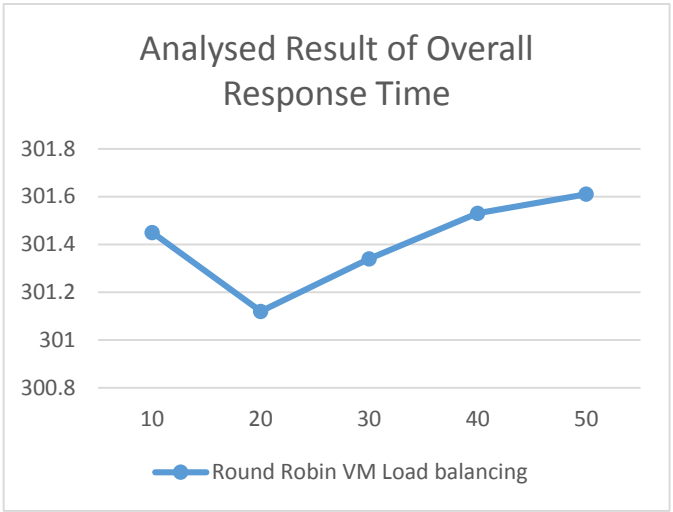


*Fig. 5 Overall Response Time*

## V. CONCLUSIONS

Analyzed result shows that Round Robin Load Balancing consumes less time for overall response time and data centre processing time over Round Robin method. When number of virtual machines increases, it takes more time for over all response time. As a future work we would like to implement more algorithms in cloud environment for resource provisioning and comparing results of different algorithms based on response time and data processing time.

## REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering IT services as the 5th utility", Future Generation Computer Systems 25 (6) (2009) 599-616.

[2] S.Suganya, P.RajiPriyadharshini, A.Mas anam, "Resource Provisioning in Cloud Computing Based on Virtual Machines" , International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 3, March 2014

[3] R. Buyya, C. Vecchiola, S. T. Selvi, "Mastering Cloud Computing", Morgan Kaufmann, USA, 2013.

[4] D. Petcu, "Multi-Cloud: Expectations and Current Approaches", Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds (MultiCloud 2013), Prague, Czech Republic.

[5] R. Buyya, R. Ranjan and R. N. Calheiros, "Intercloud: Utilityoriented federation of cloud computing environments for scaling of application services," Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010), Busan, Korea.

[6] Martin Randles, David Lamb, A. Taleb-Bendiab(2010), "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", IEEE 24th International Conference on Advanced Information Networking and Applications Workshops.

[7] Akshay Daryapurkar, Mrs. V. M. Deshmukh, "Efficient Load Balancing Algorithm in Cloud Environment", International Journal Of Computer Science And Applications, Vol. 6, No.2, Apr 2013.

[8] B. KalaiSelvi, Dr. L. Mary Immaculate Sheela, "A Survey of Load Balancing Algorithms using VM", International Journal of Advancements in Research & Technology, Volume 3, Issue 8, August-2014.

[9] Soumya Ray, Ajanta De Sarkar, "Execution Analysis of Load Balancing Algorithms in Cloud Computing Environment", International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol.2, No.5, October 2012.