Volume 5, Issue 6, June 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Trifle Scaling Algorithm for Optimal Resource Utilization in Cloud Environment

[1]N. Soundarya[*], [2]R. Brintha, [3]S. Bhuvaneshwari

[2]Research Scholar, [3]Head of the Department

[1, 2, 3,] Department of Computer Science, Pondicherry University,

Karaikal Campus, Pondicherry, India

Abstract*— Elasticity characteristic enables Cloud Computing technology to scale the resources based on the usage of the users. Many researches has been carried out to propose an algorithm for scaling so as to eradicate the downtime effective usage of the Physical Machines in the data centers. There exist many possibilities to meter extra cost if the scaled resources have been made idle for long time. In this paper, we propose a trifle approach to enable cost-effective elasticity for cloud applications. The novel approach is user side optimization that concentrates on RESOURCE LEVEL SCALING and makes the maximum utilization of the provisioned resources. The system share the resources among the VMs created for the single service of the user. Therefore it operates with the least number of resources and makes the Cloud Greener.

***Keywords* — Cloud Computing, Scaling Techniques, API, Service optimization.**

I. INTRODUCTION

Cloud computing is a model for enabling more convenient way of delivering the computing services over the Internet, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. It has been provided cheap and easy access to externalized resources. Cloud services allow individuals and business people to use software and hardware that are managed by third parties using remote locations anywhere.

Software as a Service (SaaS) is basically refers to software in the cloud. It represents the capability provided to the consumer to use the provider's applications running on a cloud infrastructure. In these applications resources are accessible from various client devices through an interface such as a web browser. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities

Platform as a Service (PaaS). It is provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure, but has control over the deployed applications and possibly application hosting environment configurations.

Infrastructure as a Service (IaaS). It is provided to the consumer is to make use of processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications.

There is a steep increase in the popularity of Cloud computing technology. The major reason of this popularity is the user's ability to scale their resources depending on the demand of their service. The approach is typically based on charging the application owners for using one or more Virtual Machines (VMs) per unit time. These VMs are then hosted for execution on different Physical Machines (PMs) at the infrastructure provider's cloud. In addition, many applications hosted in a cloud environment may have varying resource demands as they are expected to serve wide range of workloads, some of which could be predictable while others fluctuate based on using the application. For example, an e-commerce website may have a workload fluctuation during the release of some promotion activities or other events. Within this context, on demand scaling (also known as dynamic or online resource provisioning) of applications becomes one of the most important features of cloud services. This feature enables real-time acquisition/release of computing resources to adjust the applications performance in order to meet QoS (Quality of Service) requirements, while also minimising the infrastructure provider's operational cost to remain competitive.

II. RELATED WORK

Application scaling has been focused on transforming performances target into underlying resources. It classifies an application into multiple service provision each user with enough resources to meet the application's peak workload.
In contrast, clouds focus on providing metered resources on demand and quickly scaling applications up and down whenever the user demand changes. Therefore, are needed to address challenges brought by this requirement for high

elasticity. For example, vendors like Right Scale require application owners to manually specify scaling rules after an application is deployed. These rules specify the upper and lower bounds of the number of servers, the conditions to trigger scaling and the number of changed servers in each scaling. The policy-based mechanism assumes that the application owners have particular knowledge of the application being executed so that they can define proper policies. However, this is not always the case.

In this context many resources focus on optimization service resource management through relevant resource scaling techniques. The dynamic level of scaling algorithm proposes a better resource technique. Resource allocation and algorithm are advocated in real time execution information. It used to dynamic resource allocation action. The cost and time model with differential evaluation algorithm would enable generating the optimal tasks schedules to minimize job completion of cost and time..

In summary, existing resource scaling management approaches the service resource utilization efficiency, whereas they have considerable limitations due to inflexibility of using limited resources metric monitor. In contrast to this paper is unique and prototype service optimization approach that serves to achieve service resource utilization efficiently. This paper focus on resource level scaling application for multiple resource provision system for greener future. It intelligent trifle scaling algorithm utilized dynamic threshold and scaling parameters under allocated to multiple server using metric monitor statistics. This scaling method is more accurate to consider the timing control and scaling behaviour.

III. SYSTEM ARCHITECTURE

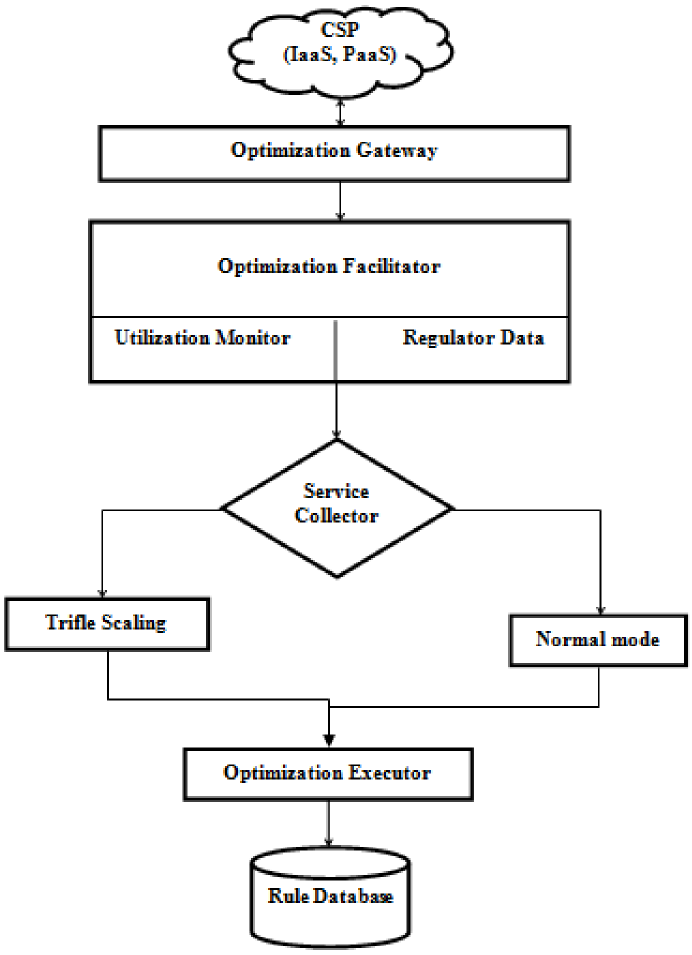


Figure 1:Trifle Scaling System Architecture

Cloud Service Provider

The CSP provide services to multiple users according to the users' requirements. This provider provides the service through optimization gateway and Application Programming Interface (API). This is the just like agreement base application between CSP and Optimization gateway.

Optimization Gateway

It is interface between the system components and service provider. It sends the service request through cloud service API. There are 3 types of service request to be send

1. General Service into for retrieving service specification, settings and status information.
2. Service utilization data request for service resource monitor data.
3. Service manipulation request which make certain changes to the services.

Optimization Facilitator

Utilization monitor monitors the workload of the service and utilization data regulator to work together handling the service metric monitors and regulating the utilization data. It supports both cloud services of scaling optimization under trifle scaling algorithm. This algorithm takes optimization rules, according to the relevant monitoring period, frequency, up/down green limits and thresholds in the optimization rule. Optimization Facilitator contains Utilization Monitor and Regulator Data. The utilization monitor calculates the speed of CPU utilization and monitors the VM which is given to the user. Using utilization monitor we can check it is green efficiency or not. It also checks threshold-up and threshold-down. The regulator data monitors the workload pattern.

Service collector

According to the regulator data the service collector checks VM size of each user. Each user having multiple services i.e. (multiple VM) is allocated .When user using the VM resources excess the workload and it reach at the peak time, that time service collector check the idle resources of the multiple services(i.e multiple VM) for each user. And resources are utilized by user efficiently. So the resources never wasted and environment became greener.

Optimization Executor

Optimization executor implements the optimization scaling and modifies the rule according to the workload pattern and also updates the new provision resources. Cloud service optimization having different scaling and threshold value to execute the optimization process.

Rule Database

It stores optimization rule relevant parameter for each optimization. A rule has 3 sections. General information that is service into rule name, id, and times. Relevant optimization monitor, green boundary and trigger parameter and notification information. And the entire rule updated in rule database.

IV. ALGORITHM

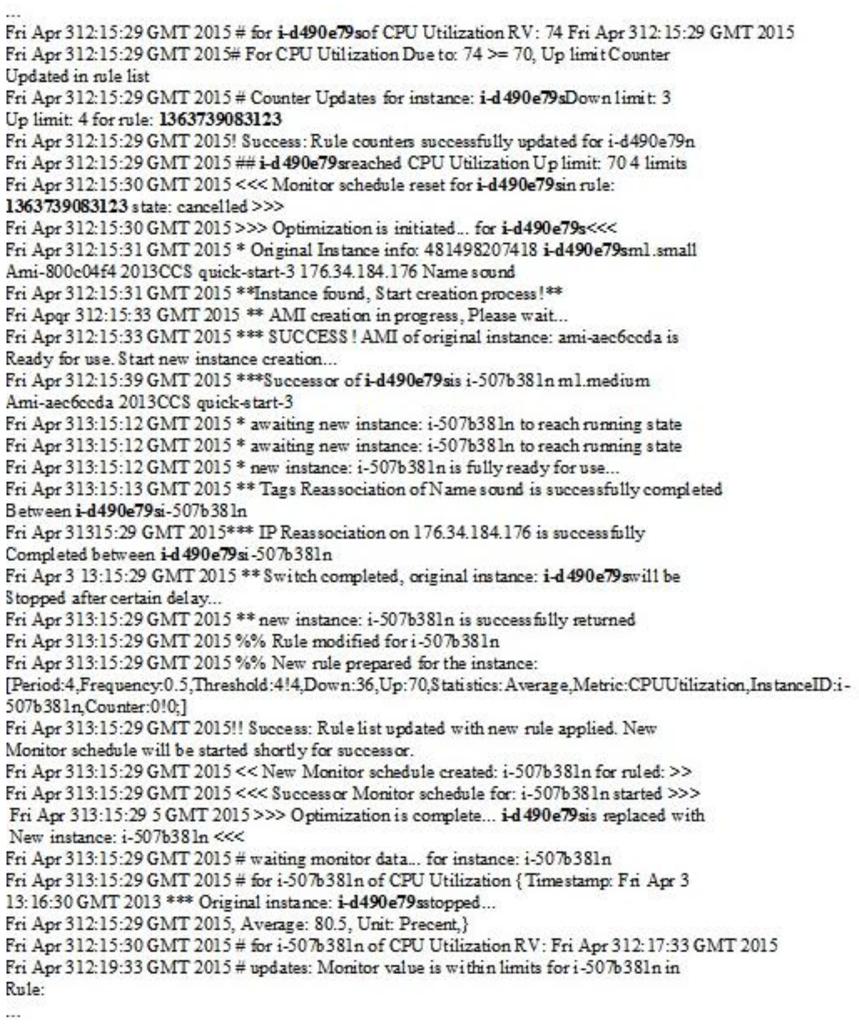
Trifle Scaling Algorithm:

```
Input get from user
Initialize SU, SUL, DC, UC, DT, UT, service collector, id appname;
Initialize all the VM allocation status to AVAILABLE in the VM state list for each user;
While (each user request are received by the service collector)
Do {
        Service collector data queue the requests;
For each SU in SUL
        If SU<DL then
        INC DC;
         End if
Else if SU>UL then
         INC UC;
End else if
         Display Notification;
End for
While DC>=DT or UC>=UT DO
Display Notification;
SET timer, with schedule to Stop
UNTIL STOP
If (service collector contain any notification current user base
&& VM allocation status == AVAILABLE)
{
Call scaling optimization. Scale with id, appname, SUL
The VM is allocated to the idle resources;
}
Else
        {
                No error occur during scaling process
        }
Display Notification
End else
End while
}
```

V. IMPLEMENTATION

In this paper user use multiple VM to optimize the resource efficiently. Each user allocated the multiple VMs to optimize the resources. First the user requested their requirements, CSP provides the resource and provider allocate for each user. According to the monitor the regulated data, service collector check the idle resources for each user at the peak time the user uses the idle resources and to optimize the resource efficiently. For example period, frequency, threshold value, metrics and notification. Here notification email will be sent automatically, when the user choose to do so. The performance of VM statistics with regards the capabilities of processing CPU intensive tasks and throughput of disk and performance of network.

…



```
...
Fri Apr 312:15:29 GMT 2015 # for i-d490e79sof CPU Utilization RV: 74 Fri Apr 312:15:29 GMT 2015
Fri Apr 312:15:29 GMT 2015# For CPU Utilization Due to: 74 >= 70, Up limit Counter
Updated in rule list
Fri Apr 312:15:29 GMT 2015 # Counter Updates for instance: i-d490e79sDown limit: 3
Up limit: 4 for rule: 1363739083123
Fri Apr 312:15:29 GMT 2015! Success: Rule counters successfully updated for i-d490e79n
Fri Apr 312:15:29 GMT 2015 ## i-d490e79sreached CPU Utilization Up limit: 70 4 limits
Fri Apr 312:15:30 GMT 2015 <<< Monitor schedule reset for i-d490e79sin rule:
1363739083123 state: cancelled >>>
Fri Apr 312:15:30 GMT 2015 >>> Optimization is initiated... for i-d490e79s<<<
Fri Apr 312:15:31 GMT 2015 * Original Instance info: 481498207418 i-d490e79sml.small
Ami-800c04f4 2013CCS quick-start-3 176.34.184.176 Name sound
Fri Apr 312:15:31 GMT 2015 **Instance found, Start creation process!**
Fri Apqr 312:15:33 GMT 2015 ** AMI creation in progress, Please wait...
Fri Apr 312:15:33 GMT 2015 *** SUCCESS ! AMI of original instance: ami-aec6ccda is
Ready for use. Start new instance creation...
Fri Apr 312:15:39 GMT 2015 ***Successor of i-d490e79sis i-507b381n m1.medium
Ami-aec6ccda 2013CCS quick-start-3
Fri Apr 313:15:12 GMT 2015 * awaiting new instance: i-507b381n to reach running state
Fri Apr 313:15:12 GMT 2015 * awaiting new instance: i-507b381n to reach running state
Fri Apr 313:15:12 GMT 2015 * new instance: i-507b381n is fully ready for use...
Fri Apr 313:15:13 GMT 2015 ** Tags Reassociation of Name sound is successfully completed
Between i-d490e79si-507b381n
Fri Apr 31315:29 GMT 2015*** IP Reassociation on 176.34.184.176 is successfully
Completed between i-d490e79si-507b381n
Fri Apr 3 13:15:29 GMT 2015 ** Switch completed, original instance: i-d490e79swill be
Stopped after certain delay...
Fri Apr 313:15:29 GMT 2015 ** new instance: i-507b381n is successfully returned
Fri Apr 313:15:29 GMT 2015 %% Rule modified for i-507b381n
Fri Apr 313:15:29 GMT 2015 %% New rule prepared for the instance:
[Period:4,Frequency:0.5,Threshold:4!4,Down:36,Up:70,Statistics:Average,Metric:CPUUtilization,InstanceID:i-
507b381n,Counter:0!0;]
Fri Apr 313:15:29 GMT 2015!! Success: Rule list updated with new rule applied. New
Monitor schedule will be started shortly for successor.
Fri Apr 313:15:29 GMT 2015 << New Monitor schedule created: i-507b381n for ruled: >>
Fri Apr 313:15:29 GMT 2015 <<< Successor Monitor schedule for: i-507b381n started >>>
 Fri Apr 313:15:29 5 GMT 2015 >>> Optimization is complete... i-d490e79sis replaced with
 New instance: i-507b381n <<<
Fri Apr 313:15:29 GMT 2015 # waiting monitor data... for instance: i-507b381n
Fri Apr 313:15:29 GMT 2015 # for i-507b381n of CPU Utilization {Timestamp: Fri Apr 3
13:16:30 GMT 2013 *** Original instance: i-d490e79sstopped...
Fri Apr 312:15:29 GMT 2015, Average: 80.5, Unit: Precent,}
Fri Apr 312:15:30 GMT 2015 # for i-507b381n of CPU Utilization RV: Fri Apr 312:17:33 GMT 2015
Fri Apr 312:19:33 GMT 2015 # updates: Monitor value is within limits for i-507b381n in
Rule:
...
```

Figure 2: System Logs

Thus the above figure demonstrated in an example of logged saved when optimization is occurs. The time needed for VM creation, booting, image-creation, is highly valuable. When successor is fully available transformation is made involving the tag and IP association is initialized when the real time green up/down using for an application with the response application running parameters along with the calculated optimal VM number, this system initiate modification request to update the capacity of application environment. Here the optimal VM size is calculated according to the period resource utilization data against its provision VM size.

VI. EVALUATION

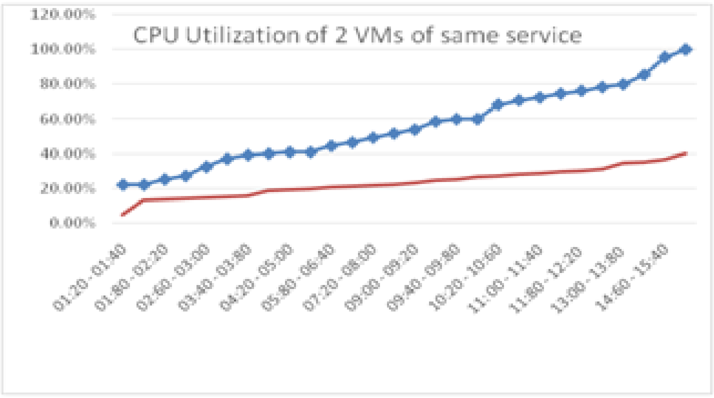


Figure 3 : CPU Utilization

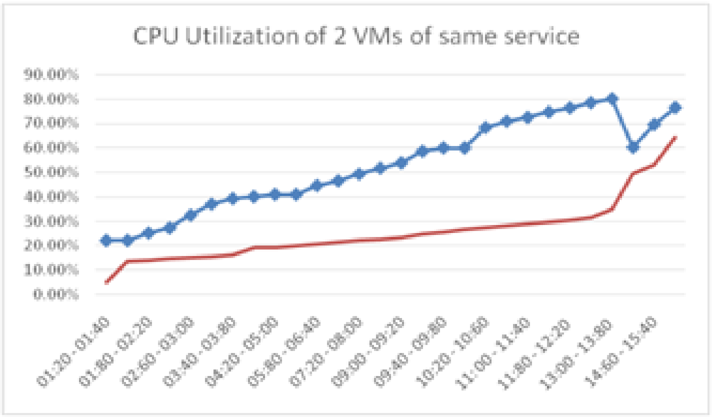


Figure 4 : CPU Utilization of VMs using Trifling Algorithm

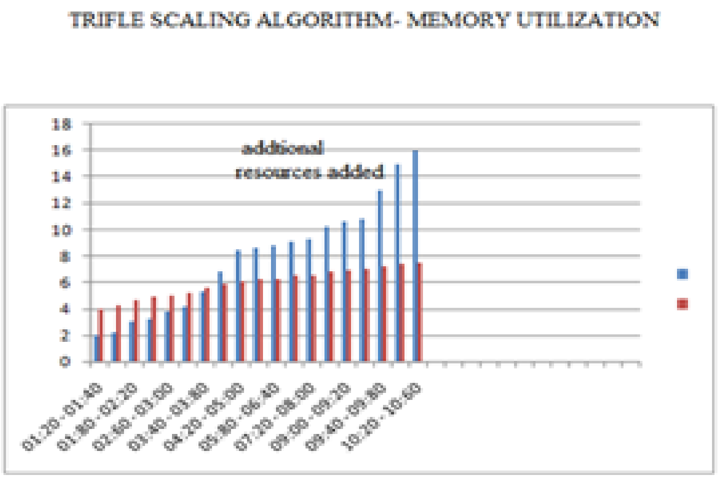


Figure 5:Memory tilization of VMs using Trifling Algorithm

A series of experiments are conducted to evaluate the effectiveness of the proposed approach. This experiment, each user has been allocated number of VMs and it allocated workloads are different dynamics. The workloads used in the experiment have an overall time of the initial unit value. it increases slowly at first and fairy dramatically until reaching the maximum value; subsequently, it decreases rapidly for the following 10 minutes and rather gradually until the end. Under such a workload, CPU utilizations of two ordinary VMs respond reasonably as the overall trend of the workload.

fig 3 shows the normal CPU utilization of 2 VMs services. Then fig 4 show that CPU utilizations of 2 services using trifle algorithm. Fig5 shows that memory utilization of 2 VMs of same services. Initially the first VMs is allocated to the user. By monitoring the usage of user to check the threshold limit and it goes periodically. When it reach the threshold limit to added the other service of idle resources to be given to the first service. Likewise the process goes on. Here second service is requested of 16 GB where has been provided. the threshold limit set as 1.6!3.2.the service consumed 12.2 GB of memory in 9 time. Therefore the need more spaces so it add in idle resource of first services and threshold limit automatically updated. By this evaluation it proved at the memory utilized efficiently by providing the resources according to the usage of services

VII. CONCLUSION

In this paper, we found an issue that the traditional method of VM level scaling leads to over estimation and over provisioning of resources and increase the CPU Utilization so as to meet the requirements of the user. To overcome this issue, we proposed a trifle approach that operates resource level scaling and enhances resource utilisation for cloud service operations. This intelligent platform based on the trifle scaling approach inherently reduces the cost on both the sides: Cloud Provider as well as Cloud Consumer. The efficiency of the proposed model has been tested successfully using the standard benchmarks of Amazon Ec2 instances.

REFERENCES

[1] Rui Han, Li Guo, Moustafa M. Ghanem, Yike Guo*, "Lightweight Resource Scaling for Cloud Applications", CCGrid 2012.

[2] [9] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "EnaCloud: An Energy-saving Application Live Placement Approach for Cloud Computing Environments", *IEEE International Conference on Cloud Computing*, pp. 17-24, 2009

[3] Amazon.com, “Amazon Web Services (AWS),” Online at http://aws. amazon.com, 2008.
[4] www.wikipaedia.com
[5] E. Feller, L. Rilling, and C. Morin, "Energy-Aware Ant Colony Based Workload Placement in Clouds", *Grid Computing (GRID), 12th IEEE/ACM International Conference,* pp.26,33, 21-23, 2011
Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "CloudScale: elastic resource scaling for multi-tenant cloud systems," in SOCC'11, Cascais, Portugal, 2011