International Journal of Research in Advent Technology, Vol.7, No.3, March 2019
E-ISSN: 2321-9637
Available online at www.ijrat.org

Role of Management Centre in Multi Cloud Computing System

B.J.D Kalyani[1], Dr. Kolasani Ramchand H Rao[2],
Research Scholar, Department of Computer Science and Engineering, Acharya Nagarjuna University[1],
Principal, ASN Degree College, Tenali[2]
Email: kjd_kalyani@yahoo.co.in[1], ramkolasani@gmail.com[2]

Abstract- Multi cloud computing enables the enterprises for effective management of their workloads. Now-a-days multi cloud computing flourishes in providing best of breed services to the end user. Management centre is necessary to design, deploy, control and monitor multiple clouds in order to carry out enterprise workloads. This paper proposes architecture for multi cloud computing system and describes the functionality of multi cloud management centre and also provides echo-efficient policies for the reduction in energy conservation.

Index Terms-Multi Cloud Computing Sytem1; Multi Cloud Management Centre2; Eco-Efficiency.

1. INTRODUCTION

The term multi cloud [1] refers to a use-case in which a business implements multiple different services, platforms, and applications into its cloud architecture. Rather than using a hybrid cloud, a public cloud, or a private cloud, the business merges several different clouds into one complete platform. Enterprises are generally using MCCS to store secure data in multiple private clouds and prefer public clouds for less secure data. Industry experts have maintained for some time that multi-cloud is the future of cloud computing within enterprise [2].

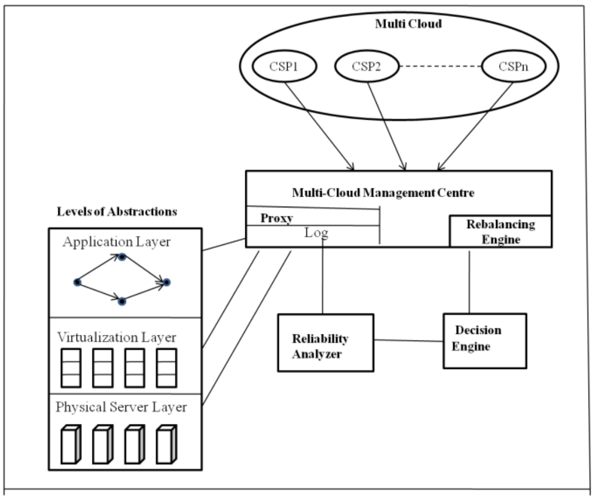
2. AECHITECTURE

Most of the computer applications are generally mobile applications; web applications are not restricted to any geographical boundary. The users of the applications are from different geographic locations. In this situation if one wants to expand globally, then MCCS are a boon. Each user located at far flung locations requires high performance not only for certain period of time but a consistent performance is required. With single cloud system this task cannot be achieved. The solution is implementation of MCCS. High performance [3] cannot be achieved not only by a single CSP but a judicious assimilation of many CSPs. A conceptual architecture is proposed to deploy and manage multiple clouds as in figure 1.

From the architecture the major components are identified as:

1. Multi Cloud Management Centre with proxy and Rebalancing Engine
2. A level of abstractions consists of Application Layer, Virtualization Layer and Physical Server Layer.
3. Reliability Analyzer
4. Decision Engine

This paper focuses on the vital component of proposed architecture Multi cloud management centre (MMC) to describe its functionality.



3. MMC

MMC is the basic building block of the architecture that provides fundamental functions that includes VM allocation, consolidation, optimization and rebalancing of user preferences. The MMC act as an interface between proxy [4] and bottom layers using a Network node consists of a large pool of cloudlets [5] that encompasses through a queue of nodes for allocation of VM and resources. The proposed architecture can be implemented, monitored using multi cloud platform Cloudify [6] placed on the above of OpenStack[7] IaaS cloud and the interactions and operations performed by MMC as

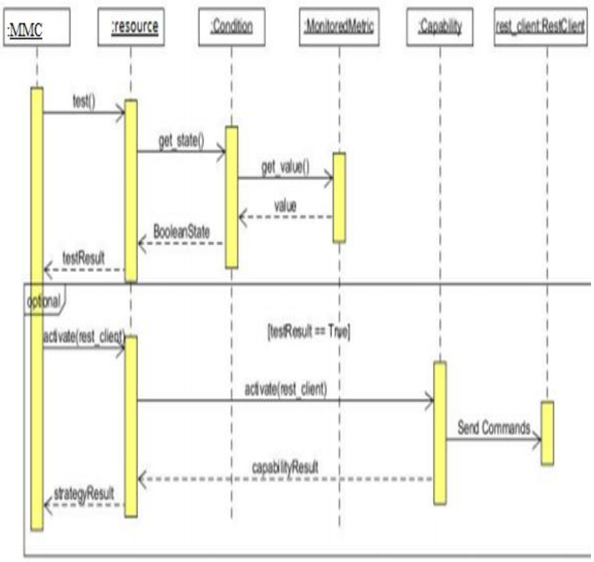


Figure 2: Sequence Diagram of MMC operations

A sample application pet clinic is deployed using the proposed architecture successfully. This paper considered reliability as metric for evaluation of functionality of components of the proposed architecture and the reliability is analyzed by the Reliability Analyzer (RA) of the proposed architecture. RA uses the following algorithm for assessment of reliability by gaining information form log [8] after the consolidation of levels of abstractions.

Algorithm: Reliability Assessment

```
Procedure
CalculateReliability(RF,nodestatus,maxRel,minRel)
Begin
Initially rel:=1, n :=1
        For I = 1 to nc
                if nodeStatus =Pass then
                        rel := rel + (rel * RF)
                        if n > 1 then
                        n := n-1;
                else if nodeStatus = Fail then
                        reliability := reliability –
(rel * RF * n)
                        n := n+1;
                if reliability >= maxRel then
                        reliability := maxRel
                if reliability < minRel then
                        nodeStatus :=dead
                        call_proc:
remove_this_node
                        call_proc: add_new_node
        End For
End
```

The log of pet clinic application can be monitored for 3 days and the resource utilization, various QoS metrics of multi cloud environment through proposed architecture and reliability is displayed by Cloudify as in figure 3.

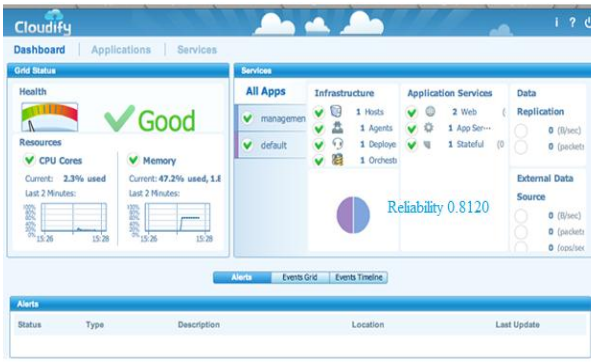


Figure 3: Reliability of pet clinic application

4. ECO-EFFICIENT ALGORITHMS

Energy conservation and service reliability [9] are the major challenges of multi-cloud computing system. In order to optimize the reliability and minimize the energy conservation three policies are proposed namely Optimized Reliability (OR), Optimized Energy Conservation (OEC) and Optimized reliability and Energy Conversation (OREC).

4.1 *Optimized Reliability policy*

In Optimized Reliability policy (OR) specified in figure 4, all the VMs running tasks of every incoming Bag of Tasks (BoT) will be sorted in decreasing order according to their utilization and all the resources will be sorted in increasing order according to their hazard-rate corresponding to the current utilization.

Optimized Reliability (OR) Policy

function OptimizedReliability(R)
// Calculate the current Hazard Rate of a resource by using equation 2
for all $j \in R$ **do**
  $\lambda_j \leftarrow r_j$.calculateCurrentHazardRate()
end for
for all $j \in R$ **do**
  $R_{sorted} \leftarrow \lambda_j$.sortHazard-rateIncreasing()
end for
return R_{sorted}
end function

4.2 Optimized Energy Conservation Policy

Optimized Energy Conservation (OEC) policy has been described in figure 5 to optimize the energy consumption by the VMs [10] in the presence of failures. In this policy all the resources will be sorted in the increasing order according to their power consumption corresponding to the current utilization.

Optimized Energy Conservation (OEC) Policy

```
function OptimizedEnergyConservation(R)
// Calculate the current power consumption of a resource by using
equation 14
for all j ∈ R do
    P_j ← r_j.calculateCurrentPowerConsumption()
end for
for all j ∈ R do
    R_sorted ← P_j.sortPowerIncreasing()
end for
return R_sorted
end function
```

4.3 Optimized Reliability and Energy Conservation Policy

The aim of this policy is to maximize the reliability and energy conservation both at the same time and is described in figure 6.

Optimized Reliability and Energy Conservation (OREC) Policy

```
function OptimizedReliabilityandEnergyConservation(R)
for all j ∈ R do
    MTBF_j ← r_j.calculateCurrentMTBF()
    P_j ← r_j.calculateCurrentPowerConsumption()
    Ψ_j ← (MTBF_j) / (P_j)
end for
for all j ∈ R do
    R_sorted ← Ψ_j.sortMTBFPowerRatioIncreasing()
end for
return R_sorted
end function
```

The ratio of $MTBF_j$ and power consumption, P_j corresponding to the current utilization for each node has been utilized to rank the resources. All the resources will be sorted in decreasing order according to the estimated ratio. A VM with maximum utilization gets allocated to a node with highest ratio value using Algorithm 1.

4.4 Expedient Load Balancing

This ELB can be served as a baseline policy. All the VMs executing tasks associated to incoming BoTs are allocated in random order as they are arriving to the next available node based on Algorithm 1.

Algorithm 1 Dynamic Resource Provisioning and VM Allocation

```
Input: Set of Bag of Tasks, B; List of Resources, R
and Policy
Output: Set of Provisioned Resources and Allocated VMs
if (Policy == OR) then
    R_sorted ← OptimizedReliability(R)
else if (Policy == OEC ) then
    R_sorted ← OptimizedEnergyConservation(R)
else if (Policy == OREC) then
    R_sorted ← OptimizedReliabilityandEnergyConservation(R)
else
//default case for ELB policy
end if
for all b ∈ B do
    for all i ∈ T do
        vm_i = t_i.taskAssignment()
        V ← vm_i
// Calculating utilization of each VM
        u_i = l_i/l_max
        U ← u_i
    end for
// Sorting VMs in decreasing order according to their utilization
    for all i ∈ V do
        V_sorted ← vm_i.sortUtilizationDecreasing()
    end for
    for all i ∈ V_sorted do
        VM_cores_i ← vm_i.coresRequired()
        for all j ∈ R_sorted do
            if ((RC_j ≥ VM_cores_i) && (S_j ≠ failed)) then
                r_j ← vm_i.allocateHost()
                RC_j = RC_j − VM_cores_i
// Calculate VM reliability by using equation 4
                rel_ij ← vm_i.calculateReliability()
// Estimate VM power consumption by using equa-
equation 14
                pow_ij ← vm_i.estimatePower()
                if (RC_j == 0) then
                    R_sorted = R_sorted − R_sorted_j
                end if
                break
            end if
        end for
    end for
end for
```

5. RESULTS

In order to implement and to evaluate the proposed dynamic resource provisioning and VM allocation policies, the hardware configuration of more than 2000 hosts of the datacenter has been taken from Los Alamos National Laboratory (LANL) [11]data set of Failure Trace Archive (FTA). The results are extracted from OpenStack instances on an average of 50 computing cycles. All the instances are generated by using 1000 BoTs with total number of tasks ranges between 100000 to 120000. The reliability with which the application has been executed with provisioned resources as in figure 6.

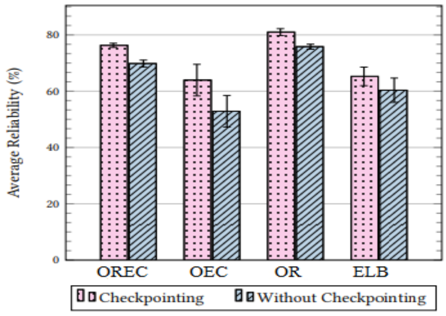


Figure 6: Average Reliability

Energy consumption encountered by the provisioned resources during the executing of application as in figure 7.

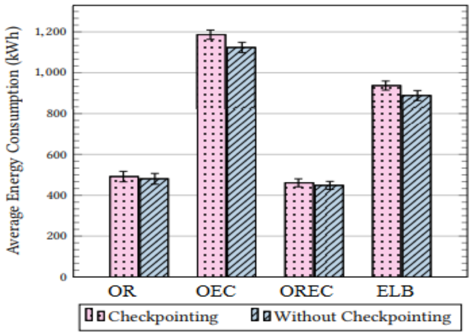


Figure 7: Average Energy Consumption

6. CONCLUSION

This paper proposes architecture for multi cloud computing system and gives an overview of the architectural components. By using the proposed architecture a sample application is deployed ane results are observed. We also proposed algorithms for eco efficiency by considering two main factors reliability and energy conservation. In future work dynamic allocation and rebalancing mechanisms of architecture are to be concentrated.

REFERENCES

[1] David Bernstein, Erik Ludvigson, Krishna Sankar, Steve Diamond, and Monique Morrow. **Blueprint for the intercloud - protocols and formats for cloud computing interoperability.** In Mark Perry, Hideyasu Sasaki, Matthias Ehmann, Guadalupe Ortiz Bellot, and Oana Dini, editors, ICIW, pages 328–336. IEEE Computer Society, 2009.

[2] Rajkumar Buyya, Suraj Pandey, and Christian Vecchiola. **Market-Oriented Cloud Computing and The Cloudbus Toolkit**, pages 319–358. John Wiley, Inc, 2013.

[3] Wenhao Li, Yun Yung, Jinjun Chen, Dong Yuan **A Cost effective mechanisem for Cloud data reliability management based on proactive replica checking** IEEE / ACM proceedings Cluster, Cloud and Grid Computing(ccgrid)2012.

[4] Y. Tamura, M. Kawakami, and S. Yamada, **Reliability modeling and analysis for open source cloud computing** , Proc. of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, vol. 227, no. 2, Apr. 2013, pp. 179–186.

[5] Amir Vahid Dastjerdi and Rajkumar Buyya, **An Autonomous Reliability-aware Negotiation Strategy for Cloud Computing Environments**, 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 978-0-7695-4691-9/12 $26.00 © 2012 IEEE DOI 10.1109/CCGrid.2012.101.

[6] https://cloudify.co

[7] https://docs.openstack.org

[8] 100 % Availability for Cloud and Data Center Applications (2013, Aug 30). [Online]. Available: http://www.cedexis.com/ .

[9] Moghaddam, M. and Davis, J. G. (2014). Service selection in web service composition: A comparative review of existing approaches. In Web Services Foundations, pages 321–346. Springer.

[10] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz. Heuristics for qos-aware web service composition. In Web Services, 2006. ICWS ’06. International Conference on, pages 72–82, Sept 2006.

[11] R. Ranjan and Anna Liu. *Autonomic Cloud Services Aggregation*. CRC Smart Services Re-port, July 15, 2009.