Decision Making Framework for Decentralized Virtual Machine Placement in Cloud Computing

Suresh B. Rathod*, V. Krishna Reddy

Abstract--- In decentralized cloud architecture, the host's configured with an autonomous local resource manager (ALRM) which takes decisions for Virtual Machine (VM) migration if it is over utilized. The ALRM takes decision for migrating its one of the VM to other peer host, by considering the peer host's utilizations received after fixed interval. This autonomous decision making results in same host identification by multiple hosts. The VM placement might results in the identified server to get over utilized and it might initiate the process of VM migration. During migration the VM and its content migrated to the identified host in plaintext form. This involves the user credentials and VM's kernel state information. Hence fault tolerance aware secure VM migration for decentralized cloud computing is introduced which avoids the over utilization of the identified server by considering its future CPU utilization, avoids the same host identification by hybrid decentralized decision making and it also ensures the VM data remain protected during migration. If failure in the decision making model the fault tolerance mechanism is introduced that helps to maintain system up for longer time. Experimental results revels that the proposed solution helps in providing security to VM's data during VM migration and avoids same destination host selection during VM placement.

Keywords--- Cloud computing (CC), Virtual Machine (VM), Controlling Host (CH), Host controller (HC).

I. INTRODUCTION

In recent years, cloud computing gaining more popularity because of its ability to provide virtualized resources. In virtualized environment VM acts as the core component, it provides uninterruptable services to end user. VM runs on top of the hypervisor and consumes underlying host's resource. Each VM differs with other VMs in the resource it consumes, including processor architecture, operating system (OS), memory type, network bandwidth and the tasks it is executing. This results in each host in DC would have multiple VM's instances running parallel with different job completion time [29]. As per NIST standard [1], virtualization addresses varying resources requirement raised by the VM's applications. The varying resource requirement by applications running on the VM gets fulfilled by the underlying physical host's resource pool. These virtualized resources assigned to the VM by the underlying hosts resource manager using partitioning, isolation, and encapsulation [1][2]. If the underlying host is unable to fulfill the resources requirement raised by the VM or the VM is consuming more resources than allocated then the underlying virtual machine manger initiates the process of VM migration from current host to the identified host. There are hypervisors like KVM, XEN, Hyper-v and ESXi provides utility functions (API) that facilitates underlying host's resource management[20].

VM migration is either the static or live migration [23].In static VM migration the VM stops its execution at origination host, VMM migrates its resources to other host. After resource migration the VM resumes its execution at migrated host. This migration type requires manual interruption. In live VM migration, the running VM instance paused at source host and resumes its execution to the destination. In live VM migration, the VM's memory, its processor state, and the network details migrated to the destination host in rounds[23]. The live migration done either in pre-copy or post-copy approach [23]. In post-copy migration, the VM's memory migrated to the destination host after the VM's processor state migrated [23]. Pre-copy migration, the VM's pages migrated to destination host, at last the processor state migrated [23].

Various authors have discussed VM migration decision making framework considering centralized or decentralized cloud architecture [29]. The cloud vendors like Google, Amazon, HP, and IBM provides cloud services to end user adopting either architecture. In decentralized cloud architectures, each host configured with ALRM which runs independent and also takes decision for VM migration by its own. To do this, ALRM referrers the neighbour host detail received at fixed interval. This own decision making by each host leads to the problem of same host identification by multiple host and over utilization of the identified host. These results in destination host might initiate VM migration.

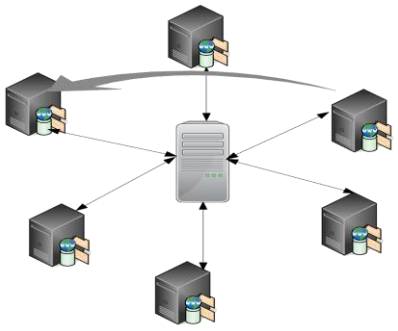

Fig.1: Decentralized cloud computing decision making framework

Manuscript received February 01, 2019

Suresh B. Rathod*, Research Scholar, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India. (e-mail: sureshrathod1@gmail.com)

V. Krishna Reddy, Professor, Department of computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India.

Cloud users do access and deploys their applications on cloud using the cloud services. Cloud user can access and managed these cloud services from remote location. Each application has varying resource requirement, this leads to extra bourdon on the underlying host. This varying resource requirement from each applications running on VM causes underlying host to be either unutilized or over utilized. The decentralized cloud architectures is shown in Fig.1, where in each host share its detail to other peer host after fixed interval. Using the peer information from the peer hosts, the ALRM running at each host takes decision for VM migration, if it finds it is over utilized due to varying resource requirement by the running VM. It migrates its running VM to other peer host using the peer information received at fixed interval. The peer in decentralized host never shares the host identification done with other peer and its future utilization after migrating its VM. This results in chances of identification of same host by multiple peer hosts. This causes sometimes the over utilization of the identified host. This requires incorporation of new decentralized framework, where the resource manager considers host's current and future CPU utilization during the VM migration and ensures secure VM migration to other hosts. This framework also ensures the decision making capability among the peer remains up by introducing the fault tolerance.

Remaining portion of this paper organized as follows: section 2 describes the related work, section 3 discusses proposed system, section 4 discusses the results of the proposed system and at last the conclusion.

II. RELATED WORK

VM's migrated to other host to avoid over utilization of the host that is caused due to excessive resource requirement by the running VM or over utilization caused due to VM placement. Various authors have discussed techniques for VM decision making.

Energy based VM placement discussed by the author in [6], here, the decision for VM migration from the host done considering penalty cost and energy consumption by the host. The solution proposed by the authors suffers with poor performance for the case when the energy cost and penalty cost increases.

Author in [7], discusses the CPU utilization based distributed load balancing on hypercube based model. Here, the individual host does takes decision for VM placement without considering the destination hosts future CPU utilization.

Optimum dynamic VM placement policy proposed by the authors in [8] works on CPU consumption by the host, authors in their work discussed maximum processing power (MPP) and random host's selection (RS) techniques for VM migration.

In [9] the author have proposed Hierarchical Decentralized Dynamic VM Consolidation Framework for VM migration, wherein they discussed how the global controller takes decision for VM migration by considering hosts future CPU utilization. The author in their work proposed the solution for VM placement using Ant Colony Optimization (ACO) technique. Again the originating host doesn't share the ant information to other, every host on overload initiates the ant to find the host with minimum utilization. There are chances to identify the same host by multiple host and initiating the migration.

Distributed load balancing using CPU utilization proposed by the author in [10] considered hypercube based VM migration. Randomized probabilistic model proposed by the author in [10], have discussed finding host's pair formation randomly and initiating VM migration in the selected host pair. The approach might skip the over loaded host during host selection, if it has more overloaded host.

Correlation based VM placement on centralized cloud architecture proposed by [12]. The approach might take more time to process the information, if the data size is large enough.

Muti target based VM placement using genetics algorithm proposed by the author in [14] considered SLA violation and CPU utilization as the parameter for VM migration decision making on centralized cloud architecture. The approach might lead to centralized failure.

In [15], authors have proposed Reinforcement Learning based VM placement wherein the authors have discussed how the centralized host learns VM deployment and puts host in sleep mode or in active mode considering the past traces.

The author [25] in his work proposed the decentralized VM migration on decentralized environment. The mechanism proposed in [25] deals with threshold based VM selection policy using upper and lower threshold limit. VM migrated to other server if its utilization reached to upper limit. There are chances that due to VM migration the selected host might be over utilized and might reinitiate VM migration.

The next section discusses the proposed decentralized predictive VM migration in decentralized cloud environment.

III. DECENTRALIZED SECURE PREDICTIVE VM PLACEMENT

This section discusses the architectural component followed by proposed predictive secure VM placement approach for the decentralized cloud environment.

3.1. The Architectural components

Here, the proposed architectural components followed by the predictive VM placement is discussed. The proposed architecture formed by incorporating distributed features like multi-tenant, distributed storage, parallel processing and multithreading [20].

Each host in the proposed architecture configured with the component shows in Fig.3.1. Here, the hosts are categorized as Controller Host (CH) and Host Controller (HC) based on the type of job they may perform. Host will termed as CH, if it does the task of providing services to end world and sharing information with HC at fixed interval. The host is termed as the HC, if it does the task of decision making and provides services to end world.

HC Resource Monitor (HCRM): This component activated when the CH acts as HC and does the task of decision making. It performs following tasks.

1. Collecting and storing peer hosts detail in current and past utilization table.
2. Providing host information to the Virtual Host Manager (VHM) as and when required.

Local Resource Monitor (LRM): This component interacts with the underlying hypervisor and does following tasks.

1. Collects underlying host detail
2. Share underlying host detail with HCRM after fixed interval.

Virtual Host Manager (VHM): Unlike HCRM, this performs following tasks.

1. Identifying source and destination during VM migration.
2. Predicting destination CH's future CPU utilization.
3. Finding upper threshold limit of the participating host.
4. Sharing next HC address with all CH.

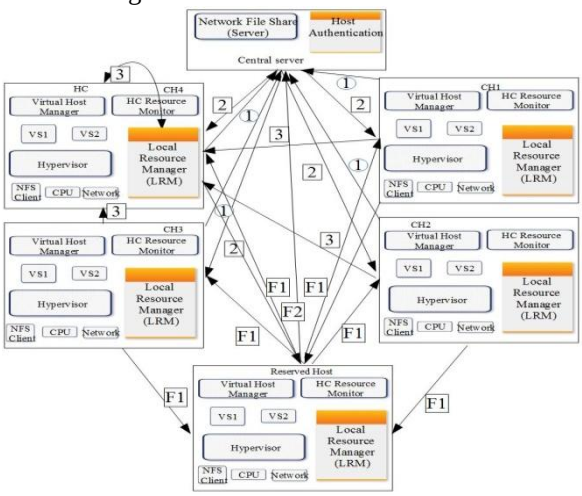

Fig. 2: Proposed decentralized hybrid host component diagram.

Initially on boot, every CH starts connecting to the central host. The central host after all CH connected, it shares the HC address with each CH. Centrla host selects one of the CH from the connected CH's and marks it as the HC. It then shares this HC address with all connected CH. The peer CH now starts sharing their own detail with the HC address after fixed interval.

From Fig.2 it found that CH1, CH2, CH3 and CH4 connected with each other forming P2P in them. Initially CH1, CH2, CH3, CH4 connects with central host as the message 1. The central host shares HC address from the CH1, CH2, CH3, CH4 by using random function (here CH4 is the HC). Central host shares CH4 address with all connected CH shown in message 2. On receiving HC address every CH starts sharing their own detail with the HC. This is shown as message 3

3.2. Proposed Predictive VM placement

Here, the predictive VM placement on decentralized cloud architecture is explored. On receiving CH detail, the VHM at HC initiates a call to DPVP to manage the VM running on various CH. The DPVP algorithm is shown in algorithm 1.

HC initiate the procedures for collecting CH detail, managing remote VM's instances, and the identification of new HC. These threads wakes up after fixed delay set by the administrator.

$$H_U = \sum_{i=0}^{n} VM_i \qquad (3.1)$$

The CH retrieves underlying CH's CPU utilization using Eq.(3.1) and shares with the HC after fixed interval.

Here H_U is the host utilization of server u. It is the sum of al lVM_i running on the host u at time interval t.

$$MAD = \frac{\sum_t^n y_t - \hat{y}}{n} \qquad (3.2)$$

Algorithm DPVP

```
1: procedure DOPREDICTIVEMIGRATION(srcAddress,minServer,hostList,currUtil)
2:     hosts ←hostList
3:     med ←med[length(currUtil) ]
4:     median ← median[length(currUtil)]
5:     vmut=findMinVMUtil(srcAddress,currUtil)
6:     setVmUtil(vmut)
7:     putil=doForecast(minServer)
8:     if oldIndex = index then
9:         doNothing
10:    else
11:        fhost=dofindFutureData()
12:    end if
13:    for each host in currUtil do
14:        if minServer == currUtil[host] then
15:            for each serv in med do
16:                if med[serv] == minServer then
17:                    address=med[serv].getKey()
18:                    if med[serv] == minServer then
19:                        medValue=med[serv].getValue()
20:                        if medValue > 0.9 then
21:                            medValue=0.9
22:                        end if
23:                        if medValue < putil then
24:                            address=FINDNEXT(srcAddress,address,fhost)
25:                            vm=findMinVM(srcAddress,currUtil)
26:                            migrate(srcAddress,address,vm)
27:                        else if medvalue > putil then
28:                            vm=findMinVM(srcAddress,currUtil)
29:                            migrate(srcAddress,address,vm)
30:                        end if
31:                    end if
32:                end if
33:            end for
34:        end if
35:    end for
36: end procedure
```

VHM calls HCRM to start retrieving active CH connections using GETCONNECTION. VHM finds source and destination CH address which has minimum and maximum CPU utilization. To do so, it uses GETMAX and GETMIN to find the minimum and maximum CH address. Upon identifying hosts, the VHM predicts the future CPU utilization of each CH and stores in FHOST(future CPU utilization). In this proposed architecture, the CH has the heterogeneous hardware configuration, static threshold might not be the good solution. CH's upper threshold computed using equation (3.2). Here, Mean Absolute Deviation MAD [9] used to find CH's upper threshold(Dynamic threshold)computed using Eq. (3.3).

Here, the y_t represent real CH's utilization and n represent a number of observations till now and $\hat{y}$represent fitted value at time t[20].

UpperThreshold=1-MAD (3.3)

After identifying CH with maximum CPU utilization, VHM searches VM with minimum CPU utilization and marks such VM for migration.

The identified VM said to be successful if it satisfies the following conditions.

- The destination CH has its FHOST lesser than the upper threshold.
- The CH's upper threshold is lesser than or equal to 0.9.

The VM placement is unsuccessful if it satisfies following conditions.

- CH's current utilization of the destination CH is greater than 0.9.
- The FHOST value of destination CH is lesser than upper threshold.

To deal with failure case, VHM starts searching new CH that has CPU utilization lesser than the current identified destination CH. The new CH selected if it has sufficient resources. The algorithm for FINDNEXT is shown in algorithm 2.

Algorithm 2 FINDNEXT

```
procedure FINDNEXT(destAddres,srcAddres,HOSTLIST)
    for each host i in HOSTLIST do
        for each host j in HOSTLIST do
            if HOSTLIST[j] ≤ HOSTLIST[j + 1] then
                temp ←HOSTLIST[j]
                HOSTLIST[j] ←HOSTLIST[j+1]
                HOSTLIST[j + 1] ←temp
            end if
        end for
    end for
    for each host i in HOSTLIST do
        if HOSTLIST[j] ≠ destAddres then
            return HOSTLIST[i]
        end if
    end for
end procedure
```

The future utilization of the CH computed using Doubles Exponential Smoothing (DES) [16]. Eq.(3.6) shows CH's future CPU utilization and smoothed value calculations.

$$s_t = \alpha y_t + (1-\alpha)(s_{t-1} + b_{t-1}), 0 \le \alpha \le 1 \quad (3.4)$$

$$b_t = \gamma(s_t - s_{t-1}) + (1-\gamma)b_{t-1}, 0 \le \gamma \le 1 \quad (3.5)$$

$$f_{t+m} = s_t + mb_t \quad (3.6)$$

Here S_t represents CH's smooth values at time t and y_t represents observed values over period t [20]. b_t represent trend factor over time period t values for the previous periodb_{t-1}. This f_{t+m}called the smoothing function [20].

3.3 Decentralized secure peer to peer VM placement

Umesh Deshpande and Kate Keahey(2015) has discussed NAS based VM migration, In their work they discussed during live VM migration VM's processor state, allocated RAM content and the data stream linked with each running task migrated to the destination host. The data stream might contain sensitive and confidential user data. In order to protect data during migration, the secure tunnel need to be established in the CH's. Tunnel formation in Linux/Unix platform achieved by SSH.

Here, the every CH configured with other CH's public key before they registers with central host. Fig.3.2 shows the SSH setup in CHs. Following are the steps to setup the SSH between CH before they start sharing information. From figure 4. The VM migration is initiated from the CH1 to CH2. Before VM is migrated it initiates the following procedure.

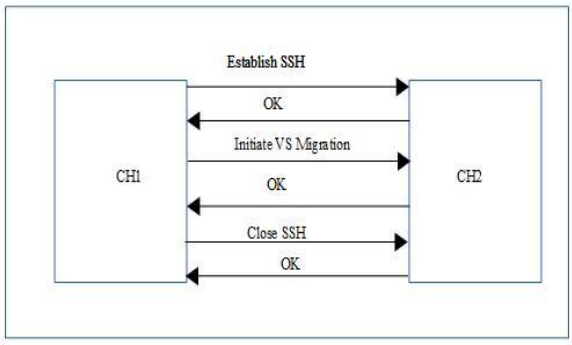

Fig.3. SSH setup in CHs

The CH1 shares its credentials with CH2. CH2 after receiving credentials does verification for the credentials sent by CH1 and does acknowledge it by ok message. After receiving acknowledge message from the CH2, the tunnel formation is initiated by CH1 to CH2. The VM from CH1 migrated to the CH2 through tunnel. On receiving VM to the CH2, it acknowledges with ok to CH1. CH1 on receiving ok gives request to close the tunnel.

IV. RESULT AND DISCUSSION

The proposed framework developed considering hybrid P2P network. The CH in current work configured with KVM/QEMU hypervisor. Java 1.6 as development platform, Libvirt, JNA, and python pandaas the supportive libraries. Every CH configured with Network File Share(NFS) client which used to access the VM's disk.The central server configured with NFS serverthat is used to store the VM's disk.

Initially all CH's starts connecting with central host. The central host takes some time to authenticate all connected CHs. Once all CHs authenticated by the central host, it randomly picks up one of the CH address marks as the HC. Central host shares this address with all connected CH. Upon receiving HC address, the CH starts sharing its detail with HC that contains host address, Number of VM, Status and CPU utilization. HC, after receiving CH details HC initiates the procedure to store the CH detail in current and past utilization table. Table 1 shows the current utilization table and Table 2 shows the past utilization table.

Table 1: Current utilization table at HC

SERVER	*CPU utilization*	*NO.VM*	*Status*
10.0.0.3	0.0804	2	FALSE
10.0.0.2	0.1227	2	FALSE
10.0.0.4	0.0811	2	TRUE
10.0.0.1	Central server		
10.0.0.5	Reserved CH		

Upon receiving HC address from the central host, CH compares its own address with the received HC address. If it finds match then, the component VHM, HCRM gets activated. The HCRM at HC starts monitoring the utilization details stored at current utilization table and does identify CH addresses for VM migration. VHM also refers same to find the next HC address.

HCRM after set interval initiates call to find CH having minimum CPU utilization and the maximum CPU utilization at current instance of time.

The VHM upon receiving the CH address starts using past utilization table and initiates thread to find the future utilization of the CH having minimum CPU utilization at current instance.

From Table1, CH with address 10.0.0.3 has maximum CPU utilization compared with 10.0.0.2 and 10.0.0.4. CH with 10.0.0.3 marked as destination CH and 10.0.0.2 marked as origination CH.

The VHM initiates DPPVP and finds future utilization of the 10.0.0.3 and 10.0.0.4. It also initiates the process to find the upper threshold limit of CH 10.0.0.3 and the CH 10.0.0.4.

The VHM compares upper threshold limit for the CH 10.0.0.3 with 0.9 and performs following checks.

- If the upper limit is greater than 0.9 it will set it to 0.9.
- If the upper limit is less than 0.9 and has its future utilization less than 0.9 then the process of VM migration initiated

If the upper limit is less than future CPU utilization then new CH identified and above two checks performed.

Before VM migration initiated, the VHM initiates a call to find the VM address that has minimum CPU utilization and marked such VM for migration. Once VM identified from the source CH 10.0.0.2 the secure channel established between CH 10.0.0.2 and 10.0.0.4.The selected VM migrated to host 10.0.0.3.

Table 2: Past utilization table at HC

SERVER	CPU utilization	NO.VM	Status
10.0.0.3	0.0804	2	FALSE
10.0.0.2	0.1227	2	FALSE
10.0.0.4	0.0811	2	TRUE
10.0.0.3	0.08	2	FALSE
10.0.0.2	0.0756	1	FALSE
10.0.0.4	0.0936	3	TRUE
10.0.0.3	0.1014	2	FALSE
10.0.0.2	0.0756	1	FALSE
10.0.0.4	0.0936	3	TRUE

The proposed approach does task to maintain the CH's utilization less than its maximum upper threshold limit. The CH said to be in normal state, if it's upper and current utilization lesser than 0.7. CH said to be over utilized if it's current and upper threshold CPU utilization greater than 0.7.

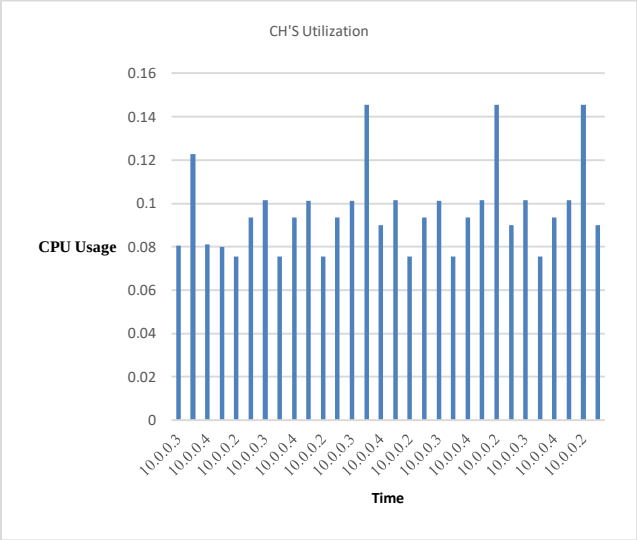

Fig.4: CH's utilization of server

Fig.4 shows CH's utilization and Fig.5 shows the Number of VM running on each CH. From Fig.4.1 and Fig.5 it is observed that the CH's utilization gets reduced due to migrating the VM from. The CH with address 10.0.0.2 has maximum CPU utilization and 10.0.0.4 has minimum utilization, so the VM from 10.0.0.2 migrated to the 10.0.0.4.

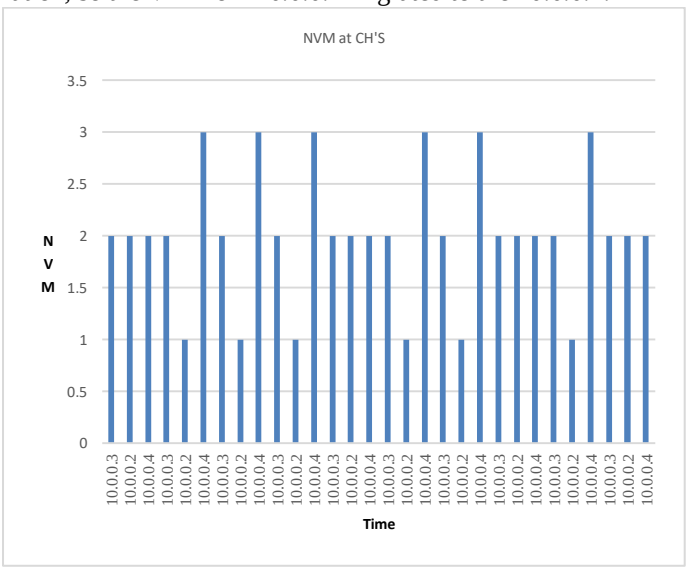

Fig. 5: No of VM on CH's.

From Fig.5 it is observed that the VM after initial migration there is no change in number of VM running on the CH 10.0.0.3. The CPU utilization CH 10.0.04 and 10.0.0.2 is changing so there is continuous VM migration in them. The VM is migrated to them as they have their future utilization lesser than 0.9.

Comparing proposed system with existing systems, the proposed system considers destination host's utilization. The ALRM activated only when the CH acts as the HC. This results in avoiding same CH identification by multiple host during VM placement. The proposed architectur has the capability to work in failure. To deal with HC failure a reserved host is added which helps to transfer the deceison making capability among the peer hosts in proposed architecture.

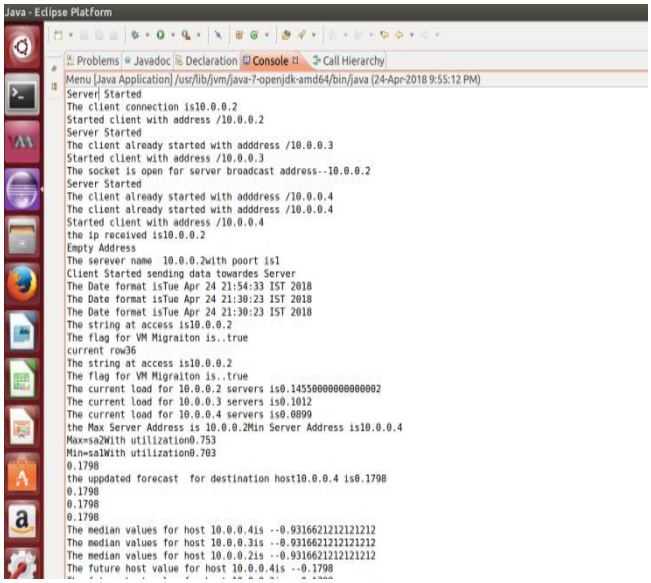

Fig.6: HC output window

To do so, after fixed interval set by the administrator, if the peer server does not receive any response from the HC after fixed interval, each server starts connection to the reserved host. From the Fig.3.1, if the CH, CH4 goes down, all the remaining CHs starts connecting to the reserved host address configured with them. Upon receiving CH's details, the reserved host finds the next HC address and shares this HC address to all CH's. After broadcasting, the reserved host updates the new HC address to the HC list for further reference.

Fig.6 shows the HC output window after applying the fault tolerance.

V. CONCLUSION

The host categorization into CH and HC avoids same host identification by multiple hosts. The proposed approach secures VM's data by establishing secure channel during VM migration process. Proposed approach restricts message exchange in HC and CHs leads to preserving message exchange in every CH, this results in less bandwidth consumption for message exchange. Fault tolerance mechanism incorporation helps in avoiding HC's failure. Prediction by DES smoothing ensures destination host never get over utilized due to VM placement and avoid unnecessary VM migration initiation due to VM placement at destination host.

ACKNOWLEDGEMENT

We would like thanks to the Doctoral committee and all the faculty members at Koneru Lakshmaiah Education Foundation and Sinhgad Academy of Engineering to support this work.

REFERENCES

1. "National Institute of Standards and Technology | NIST." [Online]. Available: https://www.nist.gov/. [Accessed: 29-Dec-2017].
2. E. Feller, C. Morin, and A. Esnault, "A case for fully decentralized dynamic VM consolidation in clouds", Proceedings of the conference Cloud Com 2012 - Proc. 2012 4th IEEE Int. Conf. Cloud Comput. Technol. Sci.,pp. 26–33, https://doi.org/10.1109/CloudCom.2012.6427585.
3. Cloud computing dened Characteristics service levels-Cloud computing news[Online].Available https://www.ibm.com/blogs/cloud-computing/2014/01/cloud-computing-defined-characteristics-service-levels/ (25 April 2018).
4. W.T. Wen, C.D. Wang, D.S. Wu, and Y.Y. Xie, "An ACO-based Scheduling Strategy on Load Balancing in Cloud Computing Environment", Proceedings of the conference Ninth Int. Conf. Front. Comput. Sci. Technol., pp. 364–369, https://doi.org/10.1109/FCST.2015.41
5. D. Grygorenko, S. Farokhi, and I. Brandic, "Cost-aware VM placement across distributed DCs using Bayesian networks", Proceedings of the conference Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9512, pp. 32–48, https://doi.org/10.1007/978-3-319-43177-2_3..
6. R. Benali, H. Teyeb, A. Balma, S. Tata, and N. Ben Hadj Alouane, "Evaluation of traffic-aware VM placement policies in distributed cloud using Cloud Sim", *Proceedings of the conference Proc. - 25th IEEE Int. Conf. Enabling Technology Infrastructure. Collab. Enterp. WETICE 2016*,pp.95–100, http://ieeexplore.ieee.org/document/7536438/.
7. Bagheri Z, Zamanifar K.. "Enhancing energy efficiency in resource allocation for real-time cloud services,"7th Int Symp Telecommun IST 2014,http:// **DOI:** 10.1109/ISTEL.2014.7000793.
8. Ferdaus MH, Murshed M, Calheiros RN, Buyya R. "An algorithm for network and data aware placement of multi-tier applications in cloud data centers*," Journal of Network and Computer Applications*,vol.98,pp.65-8398, https://doi.org/10.1016/j.jnca.2017.09.009.
9. Pantazoglou M, Tzortzakis G, Delis A. Decentralized and Energy-Efficient Workload Management in Enterprise Clouds. *IEEE Transaction on Cloud Computing*,vol.4,no.2, pp196-209,2015. http://DOI: 10.1109/TCC.2015.2464817
10. Nikzad S. "An Approach for Energy Efficient Dynamic Virtual Machine Consolidation in Cloud Environment," *International Journal of Advanced Computer Science and Applications*,vol.7 No.9, 2016.
11. Zhao Y, Huang W., "Adaptive Distributed Load Balancing Algorithm Based on Live Migration of Virtual Machines in Cloud," *In Proceedings of 5th International Joint Conference on INC, IMS and IDC, Nexus*, pp.170-175, South korea, November 2009, http://doi>10.1109/NCM.2009.350.
12. Fu X, Zhou C.Virtual machine selection and placement for dynamic consolidation in Cloud computing environment, Frontiers of Computer Science, Vol.9 no.2,pp.322-330,http://doi10.1007/s11704-015-4286-8.
13. Teng F, Yu L, Li T, Deng D, Magouls F..Energy efficiency of VM consolidation in IaaS clouds, Journal of Supercomputing, vol.73 no.2pp782-809,http:// DOI 10.1007/s11227-016-1797-5.
14. Arianyan E, Taheri H, Sharian S(2016). Multi target dynamic VM consolidation in cloud data centers using genetic algorithm, Journal of Information Science Engineering,vol.32 no4,pp. 1575-1593,2016.
15. Double exponential smoothing Insight Central (2017)[Online].Availablehttps://analysights.wordpress.com/tag/datodouble-exponential-smoothing/.(25 April 2018).
16. Mukhtarov M(2012). "Cloud Network Security Monitoring and Response System," International Transactions on Systems Science and Applications, vol.8 no.3,pp.181-185, sai: itssa.0008.2012.020,2012.

17. Anala M.R., Kashyap M., Shobha G.(2013) Application performance analysis during live migration of virtual machines, Advance Computing Conference (IACC), 2013 IEEE 3rd International,pp.366-373,Mysore,India, February 2013.
18. Tavakoli, Zahra,Meier, Sebastian,Vensmer, Alexander.A framework for security context migration in a firewall secured virtual machine environment, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),7479 LNCS,41-51.
19. Suresh B. Rathod and Vuyyuru Krishna Reddy (2018).Decentralized Predictive Secure VS Placement in Cloud Enviornment, Journal of computer science, vol.14no.3,2018.
20. Suresh B. Rathod and Vuyyuru Krishna Reddy. N Dynamic framework for secure VM migration over cloud computing, Journal of Information Processing Systems,vol.13no. 3,2017.
21. Suresh B. Rathod and Vuyyuru Krishna Reddy (2014), secure Live VM Migration in Cloud Computing: A Survey, International Journal of Computer Applications, vol.103no.2, 2018.
22. A. Shribman and B. Hudzia, "Pre-copy and post-copy VM live migration for memory intensive applications," Lect. Notes Computer. Science (including Subser. Lecture Notes Artificial Intelligence. Lecture Notes Bioinformatics), vol. 7640 LNCS, 2013, pp. 539–547, available online: https://link.springer.com/chapter/10.1007/978-3-642-36949-0_63: 28.02.2015.
23. Amazon Web Services – AWS Well-Architected Framework October 2015.
24. Michael Pantazoglou, Gavriil Tzortzakis, and Alex Delis. Decentralized and Energy-Efficient Workload Management in Enterprise Clouds, IEEE Transactions on Cloud Computing,pp.196-209,vol. 4 no.2,http:// DOI 10.1109/TCC.2015.2464817,2016.
25. Xiaoying Wang, Xiaojing Liu, Lihua Fan, and Xuhan Jia. Research Article: A Decentralized Virtual Machine Migration Approach of Data Centers for Cloud Computing, Hindawi Publishing Corporation Mathematical Problems in Engineering Volume 2013, Article ID 878542,2013.
26. Santosh Kumar Majhi, Padmalochan Bera. A security context migration framework for Virtual Machine migration,015 International Conference on Computing and Network Communications, CoCoNet 2015,pp.452-456, http:// DOI:978-1-4673-7309-8/15/$31.00 ©2015 IEEE 452February 2015.
27. Xin Wan, XinFang Zhang, Liang Chen; JianXin Zhu(2012).An improved vTPM migration protocol based trusted channel,2012 International Conference on Systems and Informatics (ICSAI2012),pp.870 – 875,China, http://DOI: 978-1-4673-0199-2/12,June 2012.
28. Fengzhe Zhang, Haibo Chen. Security-Preserving Live Migration of Virtual Machines in the Cloud, Journal of Network and Systems Management, vol.21no.4,pp.562-587, DOI 10.1007/s10922-012-9253-1.2013.
29. Suresh B. Rathod and Vuyyuru Krishna Reddy(2018),Decision Making Framework for Decentralized Virtual Machine Placement in Cloud Computing, International Journal of Engineering and Technology,vol7no.2.7,2018.