Original Research Paper

Decentralized Predictive Secure VS Placement in Cloud Environment

Suresh Baliram Rathod and Vuyyuru Krishna Reddy

Department of Computer Science and Engineering,
Koneru Lakshmaiah Education Foundation, Vaddeswaram, India

Article history
Received: 25-12-2017
Revised: 05-03-2018
Accepted: 21-03-2018

Corresponding Author:
Suresh Baliram Rathod
Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, India
Email: sbrathod.sae@sinhgad.edu

Abstract: Hosts in distributed cloud environment configured with Local Resource Monitors (LRM) that runs autonomously, monitors underlying host's resource usage and balances underlying host's resource usage by migrating Virtual Machine (VM) to other hosts. LRM takes decision for VM migration at fixed interval considering own current CPU usage and the other hosts CPU usage. The peer hosts unawares about the decision taken by the other hosts LRM about VS migration. As a result of this, there are chances that the same host selection from multiple hosts during the VS placement. This results into destination host to over utilized or the LRM at destination host initiates the VS migration. Several approaches have been proposed for decentralized VS placement that includes two threshold based VS placement, hypercube based VM placement, Ant Colony based VS placement. These approaches does not considers future behavior of the destination hosts after VS placement. This paper discusses the decentralized peer to peer Virtual Server Placement Approach (DPPVP) that considers host's current as well future CPU utilization for VS placement. The results shows that the proposed system avoids the over utilization of the destination host and host identification by the multiple hosts during VS placement.

Keywords: Cloud Computing (CC), Virtual Server (VS), Host Controller (HC), Controller Host (CH), Data Center (DC)

Introduction

In today's world, the Virtual Server (VS) is playing a major role in providing uninterruptable service to the end user. VS consume resources from underlying physical hosts including storage CPU and network. These resources gets deployed rapidly, provisioned and released with minimal management effort available on blog associated with Edwin Schouten. As per NIST, virtualization is considered as default technology to address VS's varying resources requirement. This VS's resource requirement full filled from the host considering partitioning, isolation and encapsulation. VS differ with other on type of CPU architecture, operating system, storage considered, network and the job it has assigned for execution. As a result, the host in DC has various VS's running parallel on same physical host.

The VS's varying resource demands raised by the different running applications on VS get resolved by migrating one of the VS instance from the current host. The VS migration involves selecting VS from one the host and migrating the selected VS to another host.

The VS migration is processed in such that the end-user remains unaware about VS migration. This VS migration is a critical task; hence the resources assigned to the VS need to be efficiently managed. Hypervisors like XEN, KVM, Hyper-v and VMware's ESXi provide services to manage VS.

VS migration from various hypervisor achieved though static migration or Live migration. In static VS migration the VS's need to be stopped at host manually and need to be resumed at destination host. For the Live migration, the running VS instance migrated to the destination. Here, the VS pauses its execution for a while and resumed after migration at destination host. If unsuitable physical host or VS instances selected for migration, causes unwarned effects in the future. Hence, it is necessary to consider VS selection and placement very carefully.

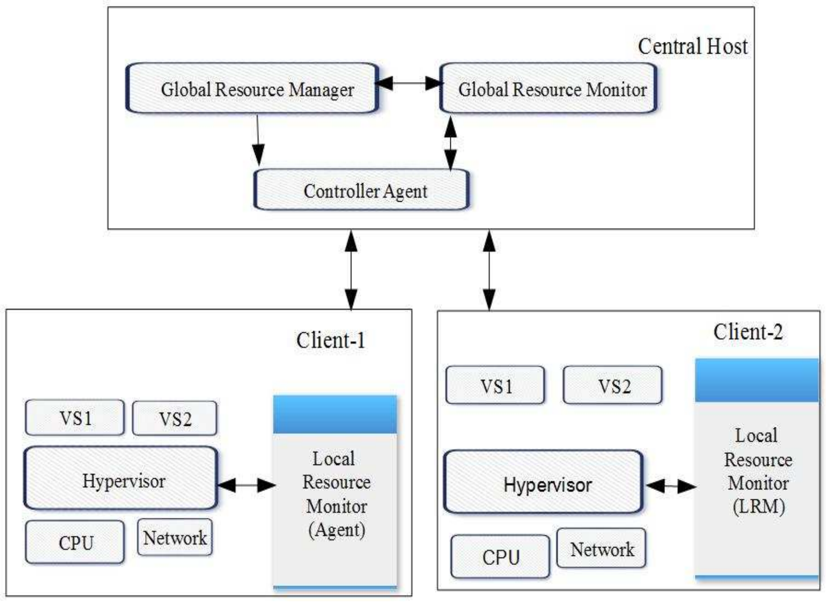


Fig. 1: Centralized cloud controlling environment

Cloud computing on the structure of organizations categorized as, centralized or a decentralized architecture. The Live VS migration is done in both centralized and decentralized cloud architecture. In the centralized cloud, central host does the task for VS selection and placement. Hosts in centralize cloud configured as shown in Fig. 1, The central host has the controller agent that communicates with all Local Resource Monitor (LRM) deployed on each client hosts. LRM collects underlying host's information and shares with the controller agent on request. The global resource monitor retrieves all client information from the controller agent. Upon receiving client details it do check client details with the threshold resource limit. If it finds violation in resource threshold limit raises alarm and shares such host detail to the global resource manager. The global resource manager then takes the decision to migrate VS from one of the client host depending on information received. All client hosts in centralized architecture managed by the single host, this can lead to single point of failure. This failure causes all services from client hosts to be down. This single point of host failure could be avoided by considering decentralized cloud environment.

The decentralized cloud architecture formed considering distributed features like multi-tenant architecture, distributed storage, virtualization, parallel processing and multithreading. Unlike in centralized controlling environment, the host in the decentralized cloud has its own controller agent that runs autonomously and does decisions for live VS migration considering neighboring hosts details. In the decentralized cloud architectures hosts shares its own resources information with other peer host at fixed interval time.

Each host in decentralized cloud configured with autonomous agent which undertakes its own decisions for VS migration. This results in selecting same hosts for VS placement by the multiple hosts. Such host selection from multiple hosts causes the selected destination host to be over utilized. To resolve this, the decentralized architecture needs to be modified such that would do decision for VS and host selection considering hosts current and future utilization of the hosts.

The rest of the paper is organized as first we discuss the previous work in decentralizing VS migration, followed proposed work and at end results of the work.

Related Work

Various authors have proposed VS placement approaches for the centralized and decentralized cloud environment. These approaches built considering linear programming, constraint programming, bin packing, ant colony and genetic algorithms. CPU utilization, no of the task, network bandwidth, disk storage, No of I/O considered as the basic parameters for the VS placement decision making.

The mechanism for host information sharing at the fixed interval was proposed by Feller *et al*.(2012).The authors have tested there approach onP2P network and have considered CPU utilization as the parameter for decision making.

Wen *et al*. (2015) discussed the mechanism for decentralized controlling and monitoring virtual resources across hosts in DC. The authors have discussed the mechanism for exchanging host information at fixed interval of time. They have considered the CPU utilization as the parameter for VS selection and placement.

Energy based VS placement approach was proposed by Grygorenko *et al*. (2016), where they have discussed VS selection considering penalty cost and energy consumption as the parameters. The proposed approach by the authors has a limitation, if the energy cost and penalty cost increased then overall performance would be decreased.

Distributed load balancing using CPU utilization is proposed by Pantazoglou *et al*. (2015), where they considered hypercube based VS placement and migration. The approach proposed by the authors did not consider whether another host in hypercube has initiated migration on the same destination host.

Benali *et al*. (2016) proposed optimum dynamic VS placement policy using CPU consumption, but they have not considered the network topology for VS placement.

Bagheri and Zamanifar (2014) discussed the Maximum Processing Power (MPP) and Random Selection (RS) approach, they have preserved same firewall rule for VS from the originating host to the destination host after migration.

Ferdaus *et al*. (2017) has proposed Hierarchical Decentralized Dynamic VS Consolidation Framework, where they discussed how the global controller does the decision making for VS consolidation.

Nikzad (2016) has proposed adaptive VS selection, where the authors have considered adaptive two-level threshold and using this threshold they categorized the hosts as troublemaker host. The authors categorized the host as troublemaker (over utilized) host. This host is categorized as trouble making at present instance or in future. The author has considered the host as input and predicted its future regarding whether the host needs VS migration.

Wen *et al*. (2015) proposed ACO based scheduling strategy; where they discussed ant colony based VS migration. The author proposed ant colony based path categorization. The authors considered positive path for VS migration. The host whose threshold limit crossed, it calls ACO to find the destination host. The authors did not consider whether there is any other host already initiated the process of migration and selected the same host as the destination during path finding. The author has considered present utilization as the parameter for VS placement.

Zhao and Huang (2009) proposed distributed live VS migration, in which the authors have considered the probabilistic pair for host selection randomly and does initiate migration between selected hosts. The random pair results in skipping over utilized host.

Fu and Zhou (2015) proposed correlation based VS placement. Here the authors has simulated there proposed VS placement using coudsim. The author has not considered the future utilization.

The next section 3 discusses the proposed DPPVP VS placement.

Proposed Work

This section discusses Requirement of proposed work, problem formulation, followed by decentralized hybrid P2P architecture, Decentralized Hybrid Peer to Peer Predictive VS placement using DES Time Series Forecasting and at end the secure tunnel formation during VS migration.

There are research works undertaken by Wang *et al.* (2013) and Nikzad (2016) which are more related to our work. Hence, before discussing our proposed methodology, a brief over view of these works is presented.

The authors Wang *et al*. (2013) in their work discussed, VM placement using unstructured P2P network. The authors have considered (src, dest, util) form to share their host utilization with other peer host. Authors have considered homogeneous host configuration and two threshold limit on resource usage. The authors have considered current host utilization as the criteria for the decision making. The VS's from the current host selected, if the VS has its current utilization greater than the (upper threshold-current utilization) of the host. Author has considered static threshold limit on each host. For the dynamic cloud environment considering static threshold is not the feasible solution. In its work the author has considered random number of neighbor host to share host information. The neighboring host changes after fixed interval.

Nikzad (2016) has proposed adaptive VS selection, wherein the authors have considered adaptive two-level threshold and uses this threshold to categorize the hosts as troublemaker host. They categorized host as the troublemaker (over utilized) at present instance or in future. The VS migration from the host initiated if the host utilization crosses the upper threshold limit. Authors has considered the host name as input and predicted its future usage. The author in their research work considered dynamic threshold computed using Mean Absolute Deviation (MAD). The author has considered 0.9 as the upper threshold for the hosts CPU utilization. Here the author has discussed hosts identification on fixed set of input.

Requirement of Proposed System

The hosts in P2P has are heterogeneous configuration shares their own information with other host. This creates extra network traffic that consumes more bandwidth towards each host and this also consumes more CPU power of the host. Each host's in

P2P takes decision for VS migration by own considering its neighboring hosts current utilization details and its threshold limit. For the heterogeneous host's configuration static limit on each host is not the feasible solution. This independent decision making with static threshold leads to chances of selecting same destination host by the multiple hosts. This results in over utilization of destination host. Because of this the destination host might initiate VS migration by own or requires shut down because of over utilization. In order to resolve this a new framework need to be established that will considers host's current CPU utilization of source host's and future CPU utilization of destination of the hosts before VS migration gets initiated.

Problem Formulation

The mapping of VS to the physical host gives the solution to the VS placement. Let C be the set of physical host represented as $C = \{CH_1, CH_2, \ldots\ldots CH_m\}$ and V be the set of virtual servers deployed on each physical server denoted as $V = \{VS_1, VS_2, VS_3\ldots,VS_m\}$. $V_{i,j}$ be the virtual server i deployed on the physical server j, such that ($1<i<n$) and ($1<j<m$). $X_{i,j}$ be the binary decision variable representing whether the VS_i selected from the host C_j. This requires VS placement to the host from the set of host C_j to be placed on one of the host from C hosts.

The mapping of V_i to the C_j such that the energy consumption of C_j at t is minimum:

$$\forall \sum_{j=1}^{m} X_{i,j} = 1 \tag{1}$$

$$\forall_j \sum_{i=1}^{m} VScpu,iX_{i,j} \le C_{cpu,j} \tag{2}$$

$$\forall_j \sum_{i=1}^{m} VSmem,iX_{i,j} \le C_{mem,j} \tag{3}$$

where, i is the virtual server and j is the physical host. Equation 2 and 3 discusses the virtual server should not exceed the physical hosts resources while placing VS on the destination host C_j.

Decentralized Hybrid P2P Architecture

The initial step for framework formation is to categorize the hosts into Controller Host (CH) or host controller HC. The host is termed as CH if it has multiple VS running simultaneously and provides services to end user. The HC works as of CH with VS decision making.

The decentralized hybrid P2P architecture is shown in Fig. 2 and its host component details described in Fig. 3. The functional detail of each host components is described below:

HC Resource Manager (HCRM)

This component is available towards each CH. It gets activated only when the CH acts as the decision maker. It performs following tasks:

- Collecting other hosts detail
- Analyzing the client hosts resource usage
- Providing host information to the VSM
- Storing CH detail to present and past utilization tables

Local Resource Monitor (LRM)

This component is available towards each CH. This module interacts with hypervisor and provides this collected information to HCRM after fixed interval.

Virtual Server Manager (VSM)

Unlike HCRM each host has VS manager that get activated only when the CH acts HC.

This component does following tasks:

- Finding the suitable source and destination host for VS migration
- Initiating VS migration
- Finding future CPU utilization of CH participated in migration
- Finding MAD values of each CH
- Finding next HC that does decision making
- Broadcasting the next HC address to all peer hosts

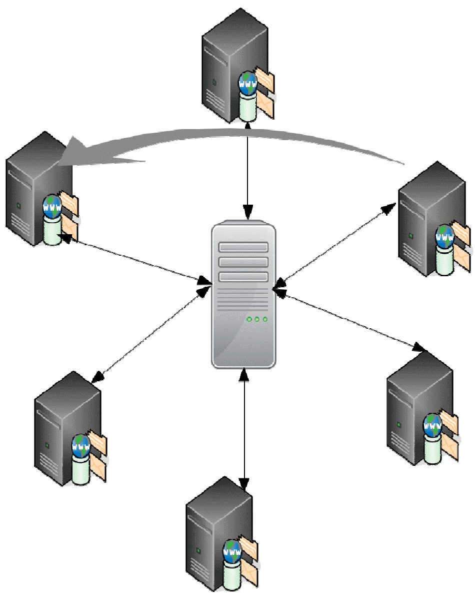


Fig. 2: Decentralized hybrid P2P architecture

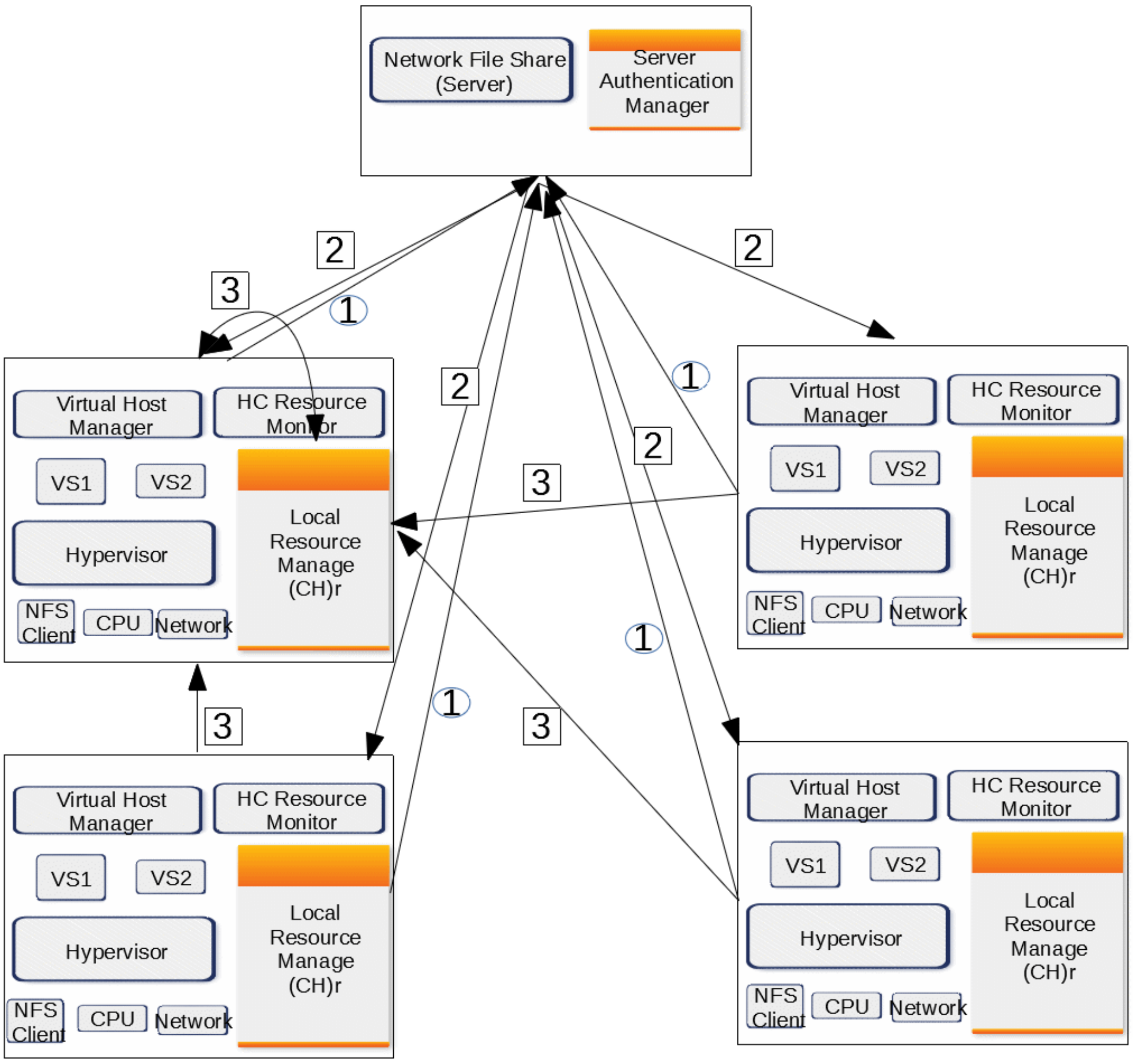

Network File Share (Server)
Server Authentication Manager
Virtual Host Manager
HC Resource Monitor
VS1
VS2
Hypervisor
Local Resource Manage (CH)r
NFS Client
CPU
Network
1
2
3

Fig. 3: CH's component diagram

Decentralized Hybrid Peer to Peer Predictive VS Placement Using DES Time Series Forecasting

Initially all CH starts connecting to the central host. Once all CH get connected, the central host selects one of the CH from the connected CH's randomly and marks as HC. This selected HC address given to the all connected CH. After receiving HC address, each CH stores this address and uses whenever it starts sharing its local detail with the HC after fixed interval.

Each CH has the running daemon threads those get wakes up whenever CH acts as HC. The HC stores all connected CHs address in active connection list. This active connection list would be referred by HC daemon threads for further processing.

The VS manager thread performs check whether the current CH acting as HC by looking out its own message structure that is shared by its LRM with HC. The CH shares its address, No. of. VS, CPU utilization and status with HC.

The HCRM at HC stores this collected CH's information in CURRUTIL that is used store the CH's current utilization and PAST_UTIL table is to stores past CPU utilization of CH.

This CURRUTIL gets referred by the HC daemon thread store; it will store all CHs detail received at fixed interval from CH's and provide this to VSM for further references. The VSM after receiving message from the HCRM makes call to the migration thread that intern gives call to the DPPVP algorithm. VSM makes the decision for VS migration by calling DPPVP algorithm shown in algorithm 1.

The HOST has all active CH connection list. This list referred by the DPPVP to find the host addresses during host identification and VS placement phase. Here, hosts has heterogeneous configuration. Equation 4 used to compute hosts upper threshold limit and get stored in MED:

$$Upper\,Threshold = 1 - MAD \tag{4}$$

Equation 5 shows the active hosts Mean Absolute Deviation (MAD):

$$MAD = \frac{\sum_t^n y_{t-\hat{y}}}{n} \tag{5}$$

Here, y_t represent actual server utilization and n represent a number of observation and $\hat{y}$ represent fitted value at time t.

Once CHs CURRUTIL received towards HCRM then DPPVP at HC initiates the procedure to identify CHs that has maximum CPU utilization and minimum CPU utilization. This CH's detail retrieved using GETMAX and GETMIN. The GETMAX provides CH that has maximum CPU utilization. GETMIN provides CH that as minimum CPU utilization.

Equation 6 used to compute the CH's CPU utilization:

$$H_U = \sum_{i=0}^n VS_i \tag{6}$$

Here H_U is the host utilization of server u. It is the sum of all virtual servers VS_i running on the host u at time interval t.

DPPVP on finding maximum and minimum utilized host addresses, it initiates the procedure for VS migration.

In this study VS placed on the destination host if its F_UTIL is less than upper threshold. Equation 9 used to compute destination CH's F_UTIL. The past CPU details fetched by referring CH's detail stored in PAST_UTIL table:

$$S_t = \alpha y_t + (1-\alpha)(s_{t-1} + b_{t-1}), 0 \le \alpha \tag{7}$$

$$b_t = \gamma(s_t - s_{t-1}) + (1-\gamma)b_{t-1}, 0 \le \gamma \tag{8}$$

$$f_{t+m} = s_t + mb_t \tag{9}$$

Here S_t represent smooth values at time t, the y_t represents observed values over a time period t. b_t represent trend factor over time period t values for the previous period b_{t-1}. f_{t+m} is objective function represents smoothed values.

Once F_UTIL value of CH get computed, the DPPVP initiates the process of VS identification at source CH and VS placement towards destination CH. The maximum upper threshold limit on CPU utilization is 0.9. CH said to be functioning smooth, if its utilization is less than 0.7. It said to be CH is over utilized, if utilization is greater than 0.7 Here we are considering max threshold value as 0.9 for each CH as of the CH's configuration.

DPPVP during the process of VS identification from the source CH do selects single or multiple VS instance.

Algorithm 1 DPPVP

```
1: procedure DPPVP(HOSTLIST)
2:     HOST = GETCONNECTIONDETAIL()
3:     MED = MED [length (HOST)]
4:     for each host i in HOSTLIST do
5:         MED[i]= FINDMEDIAN (HOST[i])
6:         CUR_UTIL[i] = HOSTLIST [i]
7:         F_UTIL[i] = FINDFUTURE (i)
8:     end for
9:     src=GETMAX (CUR RUTIL)
10:    if CUR RUTIL[src] >= 0:7 then
11:        dest = GETMIN (CUR RUTIL)
12:        putil= CUR RUTIL [dest]
13:        CUR_RUTIL[src]=CUR RUTIL[src]-Find(Min util VS)
14:    else if CUR RUTIL[src] > 0.7 then
15:        for each V S on src do
16:            dest= GETMIN (CUR RUTIL)
17:            putil= CUR_RUTIL [dest]
18:            CUR_RUTIL[src]=CUR RUTIL[src]-Find(Min util VS)
19:            GO TO STEP 22
20:        end for
21:    end if
22:    for each host i in HOSTLIST do
23:        if HOSTLIST[i] = =dest then
24:            if MED[i] >= 0.9 then
25:                MED[id] = 0.9
26:            else if MED [i] _ F HOST[i] then
27:                address= FINDNEXT (dest, src, CUR RUTIL)
28:                EstablishSSHTunnel(src,address)
29:                INITMIGRATION(src, address)
30:                return
31:            else if MED [i] _ F HOST[i] then
32:                EstablishSSHTunnel(src,address)
33:                INITMIGRATION(src, dest)
34:                return
35:            end if
36:        end if
37:    end for
38: end procedure
```

If during the decision making, the source CH has maximum CPU utilization compared with other CHs and its CURRUTIL is less than 0.7 then a single VS instance that has minimum CPU utilization compared with other VS instances, such VS marked for migration. Multiple VS instances get selected from the source CH if the utilization is greater than 0.7. VS instances get migrated to the CHs until its CURRUTIL is less than or equal to 0.7. This is shown in step 10~14.

Algorithm 2: FINDNEXT

```
1:  procedure FINDNEXT(destAddres,srcAddres,HOSTLIST)
2:    for each host i in HOSTLIST do
3:      for each host j in HOSTLIST do
4:        if HOSTLIST[j] >= HOSTLIST[j + 1] then
5:          temp = HOSTLIST[j]
6:            HOSTLIST[j]= HOSTLIST[j+1]
7:            HOSTLIST[j + 1]= temp
8:        end if
9:      end for
10:   end for
11:   for each host i in HOSTLIST do
12:     if HOSTLIST[j] != destAddres then
13:       return HOSTLIST[i]
14:     end if
15:   end for
16: end procedure
```

DPPVP after VS identification over computes the future utilization of identified destination CH using Equation 9. This destination CH's computed value compared with the MED. If the MED > F_UTIL then the selected destination CH would be discarded and new CH identified that has less utilization as of current CH. The new destination CH identification done by FINDNEXT. The address received from the FINDNEXT considered as the new destination CH address.

After finding the new host, the next step is to initiate migration between newly identified destination CH and the source CH. If the newly CH has MED < F_UTIL then the selected VS from the source CH gets placed to the destination CH.

Decentralized Secure Tunnel Formation in VS Migration

The hypervisors like XEN, VMware, KVM and Hyper-V supports live VS migration. Umesh and Keahey (2015) has discussed in their work that, in the normal VS migration the CH's in DCs hares common storages by interfacing with NAS discussed. In Live VS migration VS's processor state, its allocated RAM content and the data stream associated for each task running migrated to the destination host.

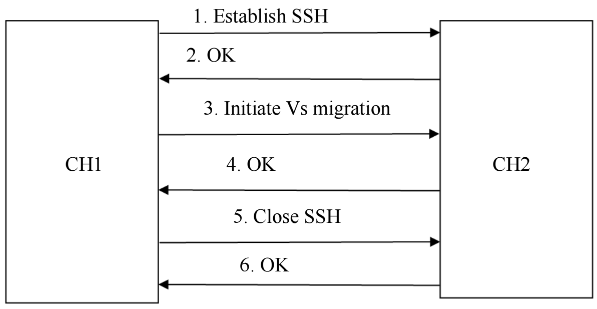


Fig. 4: SSH setup in CHs

The data stream might contain sensitive and confidential user data. In order to protect this the secure tunnel need to be established in between the CHs involves in migration. The tunnel formation in Linux/Unix platform achieved using SSH.

The network administrator configures each CH's public keys with all participant CHs. This keys would be shared by each CH's before initiating VS migration. Figure 4 shows the SSH setup in CHs:

- The CH1 shares its credentials with CH2 to establish the secure tunnel
- CH2 on verifying CH1'scredentials, acknowledges with ok message confirming establishing secure tunnel
- On receiving ok message the VS from CH1 get migrated to the CH2
- On successfully receiving VS, CH2 acknowledges with ok message to CH1
- CH1 on receiving ok gives request to close the tunnel

The next section gives detail about the result obtained from the proposed DPPVP algorithm.

Results and Discussion

The overall hardware configuration for the hosts as of in Table 1 and the VS configuration on hosts are as of in Table 2 considered while setting the proposed environment.

The CH configured with KVM/QEMU hypervisor, OpenJDK 1.6, Libvirt, JNA, python and python-panda, Network File Share (NFS) client used to share the guest image with another CH.

The central server configured with NFS server and OpenJDK 1.6. NFS in migrating helps to avoid requirement of instantiating image on the destination host. NFS keeps VS disk on NFS server and VS state migrated to the destination CH. NFS helps in reducing migration time near to zero.

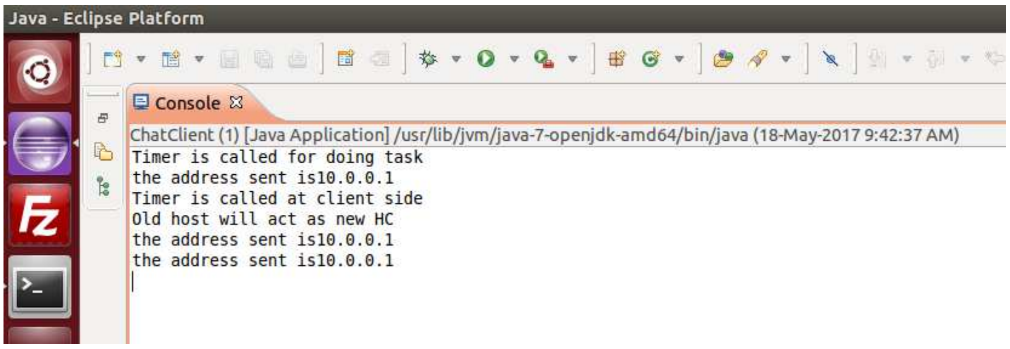
The hybrid peer to peer network formed setting one of the hosts as HC. The central host identifies HC from the connected CH's. After receiving HC address from the central server, CH's starts sharing its detail after fix interval.

Table 1: Hardware configuration of Host's

Address	Core	RAM	Operating system
10.0.0.1	I5 1st gen.	4GB DDR3,RAM	Ubuntu 14.04
10.0.0.2	I5 1st gen.	4GB DDR3,RAM	Ubuntu 14.04
10.0.0.3	I5, 6th gen	4GB DDR3,RAM	Ubuntu 14.04
10.0.0.4	I3, 5th gen.	4GB DDR3,RAM	Ubuntu 14.04
10.0.0.4	I3, 5th gen.	4GB DDR3,RAM	Ubuntu 14.04

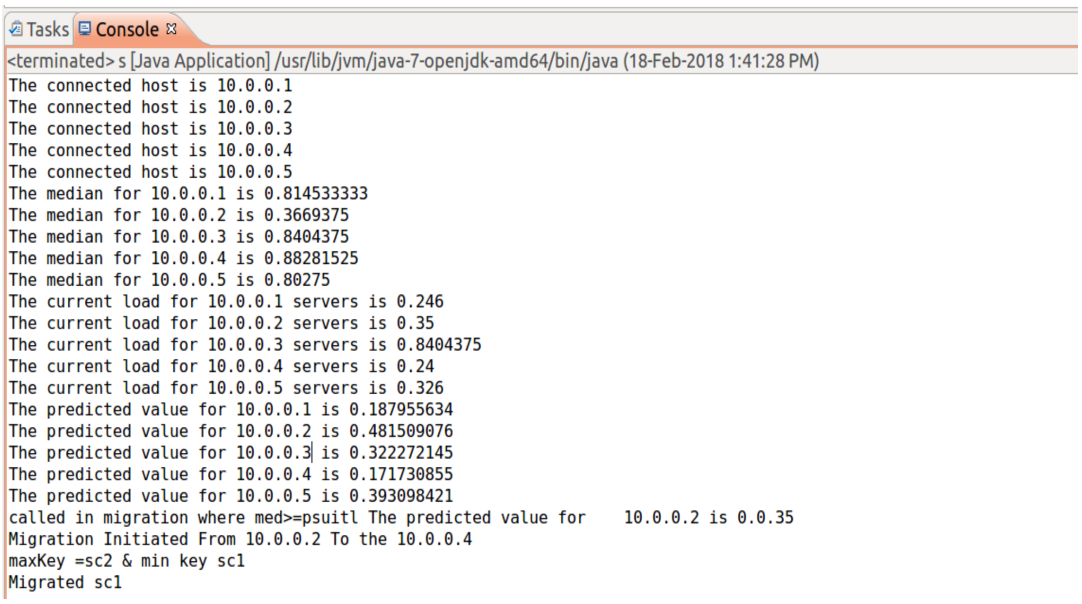
Table 2: VS configuration

Network mode	RAM	Storage	No.CPU cores
Bridge	512	1 GB	1
	1024	2 GB	1



```
ChatClient (1) [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (18-May-2017 9:42:37 AM)
Timer is called for doing task
the address sent is10.0.0.1
Timer is called at client side
Old host will act as new HC
the address sent is10.0.0.1
the address sent is10.0.0.1
```

Fig. 5: CH output Window



```
<terminated> s [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (18-Feb-2018 1:41:28 PM)
The connected host is 10.0.0.1
The connected host is 10.0.0.2
The connected host is 10.0.0.3
The connected host is 10.0.0.4
The connected host is 10.0.0.5
The median for 10.0.0.1 is 0.814533333
The median for 10.0.0.2 is 0.3669375
The median for 10.0.0.3 is 0.8404375
The median for 10.0.0.4 is 0.88281525
The median for 10.0.0.5 is 0.80275
The current load for 10.0.0.1 servers is 0.246
The current load for 10.0.0.2 servers is 0.35
The current load for 10.0.0.3 servers is 0.8404375
The current load for 10.0.0.4 servers is 0.24
The current load for 10.0.0.5 servers is 0.326
The predicted value for 10.0.0.1 is 0.187955634
The predicted value for 10.0.0.2 is 0.481509076
The predicted value for 10.0.0.3 is 0.322272145
The predicted value for 10.0.0.4 is 0.171730855
The predicted value for 10.0.0.5 is 0.393098421
called in migration where med>=psuitl The predicted value for    10.0.0.2 is 0.0.35
Migration Initiated From 10.0.0.2 To the 10.0.0.4
maxKey =sc2 & min key sc1
Migrated sc1
```

Fig. 6: Output screen when CH does makes a call for VS migration

CH shares its information in src address, CPU utilization and status from to the HC. The status flag used to distinguish CH message from HC message. Figure 5 shows the CH's output window.

Upon receiving details from CH's, HC starts storing CH's detail. This stored information will be used by the HC to do further processing. The HC crates two copies of the received information and stores in current utilization table and past table. The first table is available on local storage associated at CH and get used to identify next source and destination host during VM placement. It also used to identify next HC. The second, past utilization table is used to predict the future utilization of the hosts and is get shared among the HC. This past utilization table is stored on the NFS server.

The VSM at HC make a call to the DPPVP after 3 minutes to identify the source and destination host address for VM placement. The Fig. 6 shows the output window at HC. From Fig. 6 it found that the host 10.0.0.4 has minimum CPU utilization at current instant of time and the host 10.0.0.2 has the maximum CPU utilization. The DPPVP gives the call to find the future CPU utilization of 10.0.0.4. DPPVP also performs check whether 10.0.0.4 is suitable CH as the destination host. This check is performed by adding the VS's current CPU utilization selected for migration from 10.0.0.2 with the 10.0.0.4'scurrent CPU utilization. Upon adding VS current utilization, the 10.0.0.4'sfuture CPU utilization and upper threshold usage limit computed. In Fig. 6 the CH with address 10.0.0.4 has its future CPU utilization is 0.17 that is lesser than the median of 10.0.0.4 computed as 0.8828. As the future utilization is less than the median, the CH 10.0.0.4 marked as the destination host for VS placement. Once the source and destination CH identified the link is established in CH 10.0.0.2 and 10.0.0.4. The VS with namesc1gets migrated to the host 10.0.0.4. Table 3 shows comparative analysis of various VS placement approaches compared with DPPVP.

The proposed framework is tested considering predictive and non-predictive VS placement. In non-predictive VS placement, it found that the destination host gets over utilized or sometimes it would be under loaded. Figure 7 shows non predictive VS placement. Figure 8 shows predictive VS placement. Comparing proposed algorithm with other decentralized algorithm it found that, the proposed algorithm not only maintains source utilization at normal but at the same time it maintains the destination CH utilization at normal level. This helps in avoiding unnecessary migration from destination CH. This approach also reduce network bottleneck towards each CH by considering hybrid P2P topology and dynamic HC selection. This results reduction in bandwidth consumption to exchange host information in peer CH. As the CHs communicates with single HC, the decision for VS placement done by the single host. This helps in reducing the problem of same destination host identification by multiple host. The shared storage by NFS server and NFS client helps in reducing migration time. The proposed DPPVP approach also helps in balancing load by migrating VS between CH's in DC.

Advantages and Disadvantages

The following are the advantages of this technique:

- Destination host does not get over-utilized due to future utilization
- Peer nodes come to know which one is making the decision for VS migration
- Network bottleneck will be minimized towards each peer nodes

The following are the disadvantages of this technique:

- The decision making consumes more CPU power if the no of record is increased

Future Scope

This approach can be applied to cloud computing environment where their infrastructure is decentralized. Hence all the networks which employ decentralized approach can implement this idea. Also, various machine learning algorithms can also be applied to this idea for achieving better results. This technique can also be utilized in Software Defined Networks (SDN).

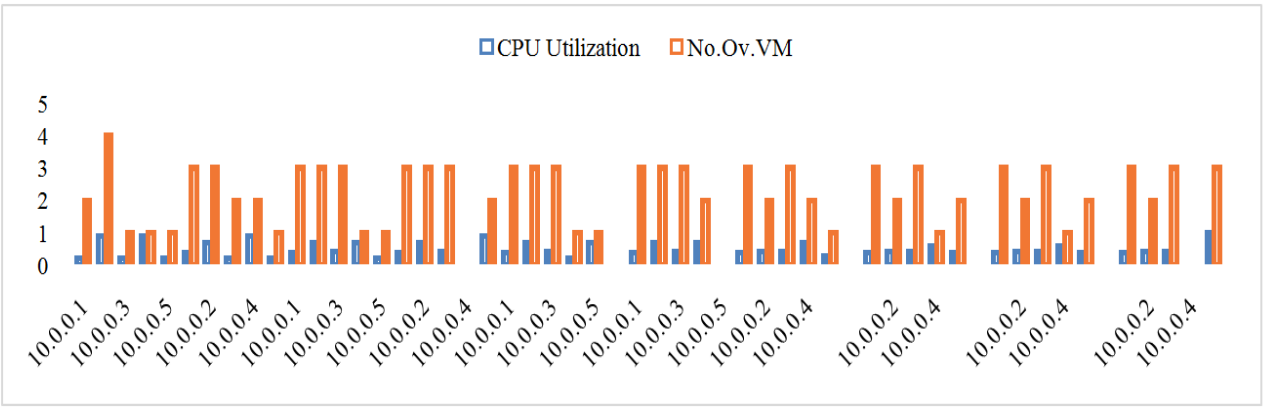


Fig. 7: Non predictive VS placement

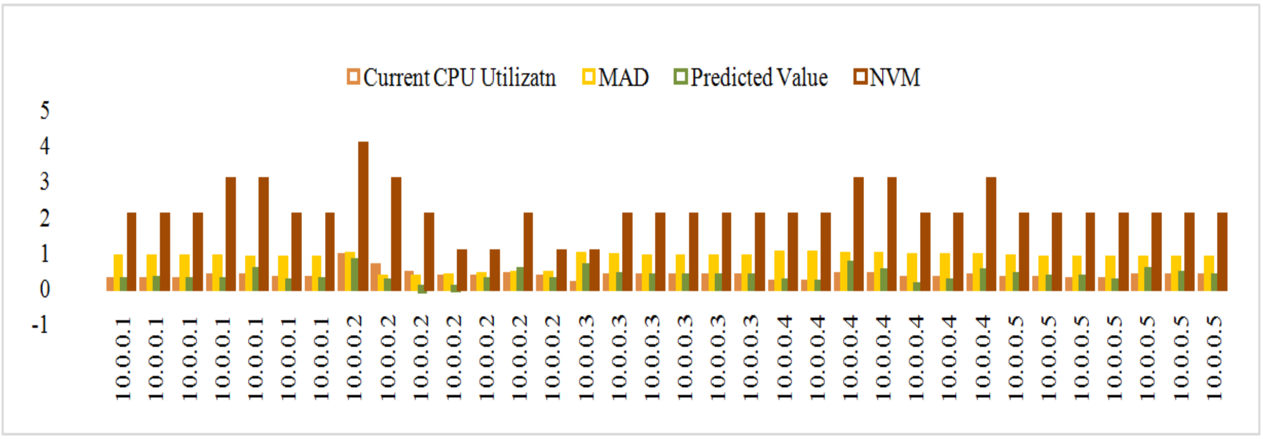


Fig. 8: Predictive secure VS placement using DES

Table 3: Comparative Analysis of Predictive VS Placement

			Parameters considered while VS placement		
Author name	Approach considered	Architecture type	Decision Making	Predictive Decision Making	Destination host information sharing
Wang *et al*. (2010)	Intel vPro technology to provide trust service to software program	Central	Central	No	No
Tavakoli *et al*. (2012)	Role based access control policies to protect against unauthorized usage of migration privileges	Central	Central	No	No
Anala *et al*. (2013)	Threat based security enforcement model using cryptography	Central	Central	No	No
Mukhtarov (2012)	Hypercube based VM placement, migration	Distributed	Peer Host	No	No
Feller *et al*. (2012)	Considered P2P VM migration.	Distributed	Peer Host	No	No
Wan *et al*. (2012)	Proposed an improved secure vTPM migration protocol	Central	Central	No	No
Zhang and Chen (2013)	Uses Firewall rule for source host and destination host authentication	Central	Central	No	No
Benali *et al*. (2016)	Proposed optimum dynamic VS Placement policy using CPU consumption	Distributed	Peer Host	Yes	No
Bagheri and Zamanifar (2014)	Discussed the maximum processing power (MPP) and random selection (RS).	Distributed	Peer Host	Yes	No
Ferdaus *et al*. (2017)	The global controller based decision making for VS consolidation.	Distributed	Peer Host	Yes	No
Nikzad (2016)	They proposed OMDD adaptive VS selection	Distributed	Peer Host	Yes	No
Wen *et al*. (2015)	ACO based scheduling strategy; where they discussed ant colony based VS migration.	Distributed	Peer Host	Yes	No
Fu *et al*. (2015)	Correlation based VS placement.	Distributed	Peer Host	Yes	No
Pantazoglou *et al*. (2015)	hypercube based VM placement, migration	Distributed	Peer Host	Yes	No

Conclusion

This paper provides the solution for VS placement in decentralized hybrid architecture. In this study the decentralized decision making policy is proposed for P2P network. Here current utilization considered for hosts identification and future CPU utilization for VS placement. The future CPU utilization consideration helps in reducing over utilization of destination host. An experimental result shows that the proposed solution maintains utilization of source and destination below threshold limit. Proposed solution also balances load across host in DC.

Acknowledgement

We would like express our thanks and gratitude towards the Head of Department Computer Science and Engineering and all the staff members of Koneru Lakshmaiah Education Foundation who have been a source of inspiration in doing this research work. Also we would like to thanks Sinhgad Academy of Engineering to support this work.

Author's Contributions

Here, in this study authors have done following contribution:

- Hosts categorization as per the role specified
- The centralized host for confirming authorized access the CHs detail information and VS's disk storage
- Restricting VS migration decision making towards HC
- Host identification considering current CU utilization
- VS placement at destination host considering destination host's upper threshold and future CPU utilization
- VS selection considering host's upper threshold limit

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and there are no ethical issues involved.

References

Anala, M.R, J. Shetty and G.A. Shobha, 2013. Framework for Secure Live Migration of Virtual Machines. Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp: 243-248. DOI: 10.1109/ICACCI.2013.6637178

Bagheri, Z. and K. Zamanifar, 2014. Enhancing energy efficiency in resource allocation for real-time cloud services. Proceedings of the 7th International Symposium on Telecommunications, Sept. 9-11, IEEE Xplore Press, Tehran, Iran, pp: 701-706. DOI: 10.1109/ISTEL.2014.7000793

Benali, R., H. Teyeb, A. Balma, S. Tata and N. Ben Hadj-Alouane, 2016. Evaluation of traffic-aware VM placement policies in distributed cloud using CloudSim. Proceedings of the 25th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, Jun. 13-15, IEEE Xplore Press, Paris, France, pp: 95-100. DOI: 10.1109/WETICE.2016.29

Feller, E., C. Morin and A. Esnault, 2012. A case for fully decentralized dynamic VM consolidation in clouds. Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science, Dec. 3-6, IEEE Xplore Press, Taipei, Taiwan, pp: 26-33. DOI: 10.1109/CloudCom.2012.6427585

Ferdaus, M.H., M. Murshed, R.N. Calheiros and R. Buyya, 2017. An algorithm for network and data-aware placement of multi-tier applications in cloud data centers. J. Netw. Comput. Applic., 98: 65-83. DOI: 10.1016/j.jnca.2017.09.009

Fu, X. and C. Zhou, 2015. Virtual machine selection and placement for dynamic consolidation in Cloud computing environment. Frontiers Comput. Sci., 9: 2095-2236. DOI: 10.1007/s11704-015-4286-8

Grygorenko, D., S. Farokhi and I. Brandic, 2016. Cost-aware VM placement across distributed DCs using Bayesian networks. Proceedings of the International Conference on Grid Economics and Business Models, (EBM' 16), Springer, pp: 32-48. DOI: 10.1007/978-3-319-43177-2_3

Mukhtarov, M., 2012. Cloud network security monitoring and response system. Proceedings of the 3rd International Conference on Cloud Computing, GRIDs and Virtualization, (CGV' 12), Nice, France, pp: 181-185.

Nikzad, S., 2016. An approach for energy efficient dynamic virtual machine consolidation in cloud environment. Int. J. Adv. Comput. Sci. Applic., 7: 1-9. DOI: 10.14569/IJACSA.2016.070901

Pantazoglou, M., G. Tzortzakis and A. Delis, 2015. Decentralized and energy-efficient workload management in enterprise clouds. IEEE Trans. Cloud Comput., 4: 196-209. DOI: 10.1109/TCC.2015.2464817

Tavakoli, Z., S. Meier and A. Vensmer, 2012. A framework for security context migration in a firewall secured virtual machine environment. 18th European Conference on Information and Communications Technologies, (ICT' 12), Springer, Budapest, Hungary, pp: 41-51. DOI: 10.1007/978-3-642-32808-4_5

Wan, X., X. Zhang, L. Chen and J. Zhu, 2012. An improved vTPM migration protocol based trusted channel. Proceedings of the International Conference on Systems and Informatics, May 19-20, IEEE Xplore Press, Yantai, pp: 870-875. DOI: 10.1109/ICSAI.2012.6223146

Wang, W., Y. Zhang, B. Lin, X. Wu and K. Miao, 2010. Secured and reliable VM migration in personal cloud. Proceedings of the 2nd International Conference on Computer Engineering and Technology, Apr. 16-18, IEEE Xplore Press, Chengdu, China, pp: 705-709. DOI: 10.1109/ICCET.2010.5485376

Wang, X.Y., X.J. Liu, L.H. Fan and X.H. Jia, 2013. A Decentralized virtual machine migration approach of data centers for cloud computing. Math. Problems Eng., 2013: 1024123X-1024123X. DOI: 10.1155/2013/878542

Wen, W.T., C.D. Wang, D.S. Wu and Y.Y. Xie, 2015. An ACO-based scheduling strategy on load balancing in cloud computing environment. Proceedings of the 9th International Conference on Frontier of Computer Science and Technology, Aug. 26-28, IEEE Xplore Press, Dalian, China, pp: 364-369. DOI: 10.1109/FCST.2015.41

Zhang, F. and H. Chen, 2013. Security-preserving live migration of virtual machines in the cloud. J. Netw. Syst. Manage., 21: 562-587. DOI: 10.1007/s10922-012-9253-1

Zhao, Y. and W. Huang, 2009. Adaptive distributed load balancing algorithm based on live migration of virtual machines in cloud. Proceedings of the 5th International Joint Conference on INC, IMS and IDC, Aug. 25-27, IEEE Xplore Press, Seoul, South Korea, pp: 170-175. DOI: 10.1109/NCM.2009.350

Umesh, D. and K. Keahey, 2015. Traffic-sensitive live migration of virtual machines. Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May4-7, IEEE Xplore Press, Shenzhen, China, pp: 51-60. DOI: 10.1109/CCGrid.2015.163