A Distributed API for Live VM Migration in Cloudlets

Videet Singhai
Sardar Patel Institute of Technology
Mumbai, India

Krushi Damania
Sardar Patel Institute of Technology
Mumbai, India

Shraddha Holmukhe
Sardar Patel Institute of Technology
Mumbai, India

Prasenjit Bhavathankar
Sardar Patel Institute of Technology
Mumbai, India

ABSTRACT
Cloud computing is an effective technology in handling computation with dynamically scalable resources. With the growth of multimedia applications, mobile applications have become resource-intensive. To provide better connectivity with the cloud, cloudlets have been introduced, which in turns provide low latency and high bandwidth. In this paper, we discuss the live migration of Virtual machines in Cloudlets using VirtualBox as hypervisor. We have proposed and implemented a distributed API Viper particularly for cloudlets, which provides users, interfaces for operating with Virtual Machines. We also test this API in different scenarios, changing RAM, CPU and CPU Stress.

Keywords
Cloud Computing, Virtualization, Live Migration, Mobile-Edge Computing, Cloudlets.

1. INTRODUCTION
The two most common IT-related terms currently in use are Internet and Cloud computing. The development of security and improvement of hardware performance led to blooming of cloud computing in the current IT industry and emerged with the foundation of virtualization technology. Cloud computing allows companies to rent their spare hardware resources to customers in a pay-as-you-go manner and eliminate hardware purchase and maintenance cost. Virtualization techniques use Virtual Machines (VM) to launch particular environments and share physical hardware resources.

The mobile devices have improved its computation power, storage, and battery lifetime but they still lack when running resource-intensive applications. Cloud-based applications are becoming popular and making such applications performance independent of mobile devices computation capacity. VM migration is a critical technology in order to protect the VM from hardware failure, for load balancing, to avoid hotspot failure [12]. It allows us to transfer the entire virtual environment to another server without affecting the applications running inside the VM. There are three approaches to migration; Hot, Cold and Live [6]. During cold migration, the guest OS is shut down before migration. Hot migration allows us to suspend the OS which will be resumed after the migration is complete whereas live migration migrates the VM while the OS is still running with a small downtime.

Live migration is one of the most promising techniques of virtualization. Live migration is performed in order to maintain the state of the migrated system same as the original with least interruption to service running inside the VM. Multiple VMs can also be migrated together in case of server failure [7].

In this article, we discuss briefly about need of cloudlets for mobile devices and live migration in Section II. Various parameters that can be used to get performance of migration are given further. That section is followed by the features of proposed API along with design architecture of the environment setup. The results are illustrated using graphs for depicting performance of migration in different test cases.

2. MOBILE EDGE COMPUTING
The number of customers using a mobile phone to connect to the Internet is constantly rising. It has long been recognized that mobile hardware is necessarily resource-poor. Mobile devices have more drawbacks compared to static clients and servers. Battery life, memory, weight, and heat dissipation are the major factors limiting the scope of the device [2].

To overcome these limitations, cloud computing was introduced where the mobile uses cloud storage and server for computation. Although cloud solves major problems and limitations of mobile devices, there are considerable delays introduced when transferring data between mobile and cloud data center. The main cloud infrastructure is located far from the mobile user. For example, Amazon's EC2 infrastructure is located only in 6 cities worldwide. When the user attempts to connect to any of the data centers, it needs to go through many network segments. This introduces unnecessary delay and can interrupt the user interaction. If the user operates real-time applications, then using clouds will be insufficient. Even in case of the Internet of Things (IoT) devices, the sensors do not have much storage and processing capacity. By the time this data, generated with high velocity is used for analysis, there could be failure or crash and the opportunity could be lost.

This paves way for implementation of cloudlet. Cloudlet is a small cloud station nearby mobile user which will be connected through LAN or WAN network to the cloud data center. Basically, the mobile user needs to interact with the cloudlet only; the cloudlet will take care of the resource management and synchronization by communicating with the data center.

A situation may arise when the mobile device moves away from the cloudlet and may lead to low performance. In such case the VM running the computation service can be migrated to a cloudlet which is nearby the mobile device. The performance degradation can be detected by monitoring the round-trip time (RTT) of the mobile device with the cloudlet server. The cloudlet should initiate a migration if it finds a nearby cloudlet which may provide a better service. It should also consider factors like VM storage, CPU usage, load and

check if it is efficient to migrate and whether the total VM migration time and downtime is in an acceptable range.

Live Migration of VM is essential to keep the application running during migration. The VM migration should occur in a seamless and abstract way without the user noticing. Pre-copy live migration is the best way for migrating the VM. The total migration time and downtime will depend on the CPU usage by the application, dirty rate and performance capability of the VM. Here, hypervisor copies all the memory pages to the destination host while the VM being run on the source host. During this process, some memory pages get updated or modified. These pages are often referred as dirty pages. These pages are re-copied to the destination host until no dirty page is left, so the transfer takes place in multiple iterations. When the entire VM is migrated along with the dirty pages, the VM on source host is stopped and the one on destination node is resumed.

2.1 Shared Storage

Shared storage is a form of repository that is shared by the two hosts between whom the migration is taking place. A VM consists of two main components, configuration or state, and VM storage space. The VM Storage is already shared between the hosts. In live migration with shared storage, only the configuration file is migrated and the storage file does not move. VirtualBox allows us to share the virtual hard disk (VDI) with another machine. For migration virtual hard disk file of the VM in source cloudlet is shared with the target cloudlet, they either use the same iSCSI targets or use storage that resides on the network and both cloudlets have access to it via NFS or SMB/CIFS. In our experimental setup, we have used NFS for shared storage. Different technologies that can be used for shared storage is Network File System (NFS), GlusterFS, NetApp Clustered ONTAP etc. NFS and GlusterFS are discussed below:

2.1.1 Network File System

NFS is one of the many network-attached storage (NAS) file system which behaves like a client/server application for a user to view, store and/or update files on a remote machine. It lets the user mount whole or a part of a file system on the server which serves as the centralized storage space. This mounted portion can be accessed by the clients based on predefined privileges. The NFS server by default executes the NFS daemon process to offer a portion of its attached disk storage to NFS clients, to trade TCP/IP packets with. The server determines what to make available to the authorized client by mentioning the parameters and directories in the configuration file. The NFS client requests access to these files by using mount command this command asks the server for an open port and if permission is granted it can view the files and directories on the server with predefined privileges [9]. In Virtual Machine (VM) Migration, NFS is used to share VM images with other KVM hosts. It can be used only to run small installations and for few clients. File locking is not supported in KVM.

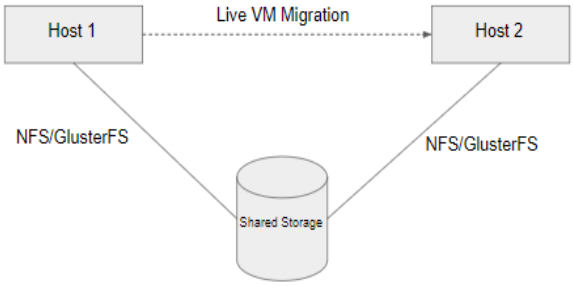

Fig 1: Shared Access Storage

2.1.2 GlusterFS

The faults of CPU, memory or network are easy to recover from while that of disk images causes problems to bring the system up and running. For disk image to be fault tolerant, GlusterFS is an online file system with petabyte level scaled-out storage using client/server architecture. If many clients perform read-write operation simultaneously on the NFS server, the load on the server increases and it also starts falling short of resources. Therefore, the idea of combining memory and power of multiple servers to form a volume to be accessed by clients arises. GlusterFS is a distributed storage system and hence it aggregates storage servers through TCP/IP or InfiniBand RDMS technology to form one enormous parallel network [10].

Servers are established as storage directory and they run a GlusterFS daemon to export a local file system as a volume. All the client-side operations can be performed by applications using standard IP networks. The client creates virtual volume from multiple remote servers and is mounted via the FUSE (File System in User Space) mechanism.

2.2 Without Shared Storage

Migration without shared storage or traditionally called Shared Nothing Live Migration allows the user to move a virtual machine even if the VM's file system resides in a storage not shared by both the hosts. During migration, the configuration file is moved first then the VM's storage file is migrated. To improve the performance, the VM's state and storage still run on the source host till the storage file is migrated and the VM starts up on the destination host.

3. VM MIGRATION IN CLOUDLETS

Initially, when the user starts the mobile application, it gets connected to the cloudlet that is present in its proximity and is nearest to it. The cloudlet sends a request to the cloud center for that user's required VM. The VM required by the user is brought to that cloudlet from the data center. All the services provided to the user are maintained by that cloudlet. As this cloudlet is nearby to the user, the user's experience with the application is very smooth. There often arises a situation, when the user is mobile, the distance between the cloudlet and the user will start increasing. There may be a point where there is a considerable delay in the connection between the user and cloudlet. Hence, the user needs to be connected to a new cloudlet which is nearest to it. Along with the new connection, the storage and services also need to be brought to new cloudlet without disturbing the performance of the application. In short, the VM of the user in previous cloudlet needs to be migrated to the recent nearby cloudlet. This migration needs to be a live VM migration since the user should not experience any downtime, although in practical world there is a downtime of few milliseconds.

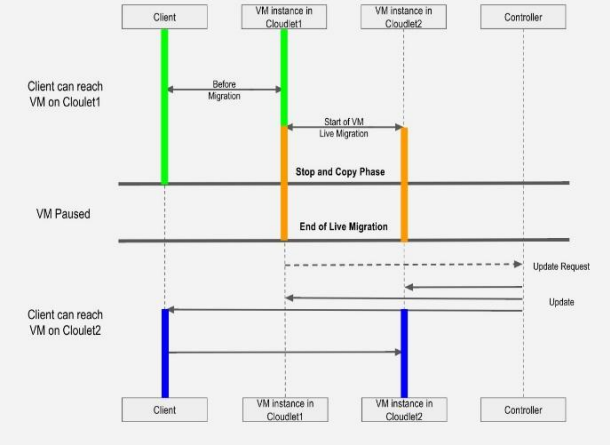

Fig 2: Detailed migration process between cloudlets

4. PERFORMANCE PARAMETERS

VM Migration has to be fast, efficient and with minimal side effects to give an overall good performance. Depending on different migration mechanisms their performance differs in metrics as optimization ways change [4,8].

4.1 Metrics

In this section, we discuss different metrics that can be considered during live migration [3].

4.1.1 Total Migration Time

It is the total time taken during every process from the start to the end of VM migration. Its value can vary depending on how much data is being moved during migration. Total data transferred from source to target cloudlet and allocated bandwidth decide this parameter.

4.1.2 Downtime

The time for which the service becomes unavailable during the VM migration is called the Downtime.

4.1.3 Page Dirty Rate

It is the rate at which the memory pages are updated by the VM. During the migration, every page must be updated as quickly as possible to maintain the consistency and to confine the total migration time and downtime

4.1.4 Link Speed

The bandwidth allocated to the link is inversely proportional to the total migration time. The faster transfer requires more bandwidth; hence it takes less total migration time.

4.1.5 Network Traffic

It is the amount of data sent from source to target cloudlet is a factor in deciding cost of migration.

4.1.6 Service Degradation

The application running inside the VM which is accessed by the mobile device gets affected during migration and service degradation is a measure to indicate the degree of that. It can be measured by the changes in throughput, response time, etc of the service.

4.1.7 Network Bandwidth Utilization

This metric is defined by the considering network traffic and migration time. Utilization is at its best when network traffic and migration time is lowest.

4.2 Overhead

Migration has its drawbacks and some performance degradation factors are discussed below [8].

4.2.1 Computational Overhead

Migration processes use up computational resources at other source and destination. Optimization techniques like data Deduplication, compression also add to it.

4.2.2 Network Overhead

Reading and writing from source to destination consume I/O bandwidth and migration process competes with other VMs as it is very network-intensive.

4.2.3 Space Overhead

It's least valuable compared to other two overheads. Techniques like snapshotting take up introduce space overhead

5. PROPOSED API

There are two types of APIs we developed -Viper Distributed REST API and Viper Python Library (package).

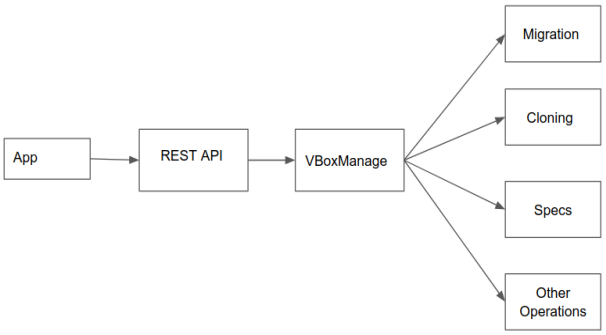

Fig 3: Workflow of Viper API

The REST API is built with Flask. The API needs to be downloaded on every system (cloudlet) that is a part of the Fog. After the package is downloaded, the programmer just needs to set details like IP address in the config.py file and the server setup gets completed. After the API is installed and configured in all the cloudlets, the database of the cloudlets need to be updated. Every cloudlet should have the metadata about its neighbouring cloudlets. Whenever a cloudlet is added in the system or deleted from the system, the database gets updated. In our Experiment, we used MySQL DB. Companies who have their own cloud center have a dedicated REST API or any other kind of mechanism that triggers certain events which in turn take care of the cloud infrastructure. The application (app) that the company runs communicates with the REST API. The API realizes the essential data and parameters of the corresponding app and performs certain activities like VM migration or load balancing. REST API gives an advantage because the app can communicate with it irrespective of the language it is written in. This API is also useful for IoT devices which need to send data on the fly for analysis.

As we can see from the above diagram, the app sends a request to the Viper REST API. The API takes care of communicating with the Virtual Machines. It provides functions like migrating, cloning, initializing a VM.

In some cases, companies prefer to use their own REST API for monitoring their cloud center. We have developed a python library for such scenarios. A programmer can just import the Viper modules and use the provided functionalities. One can create its own REST API using this python package.

6. IMPLEMENTATION

6.1 Assumptions and Constraints

Following are the assumptions and constraints

- The cloudlets are connected in VPN over LAN.
- The cloudlets are nearby each other and hence RTT latency is simulated
- Cloudlets are assumed to be performing as customer edge and are one hop away from the provider edge
- Maximum of 2 interfaces can be connected to the mobile

6.2 Architecture

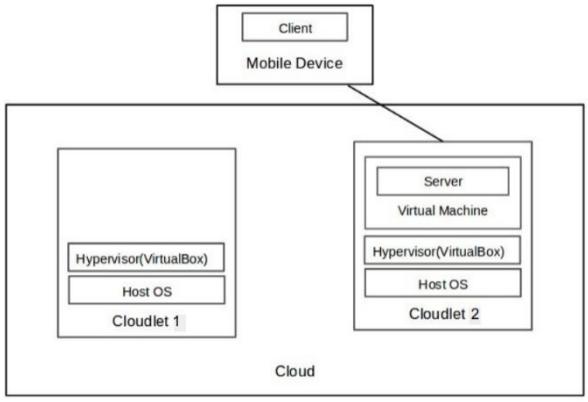

Fig 4: System Architecture

In the system, there are 2 main modules the cloud and mobile device. Cloud is VPN of cloudlets over LAN. The cloudlets are nearby each other and hence they are assumed to be a hop away from the mobile device. The cloudlet consists of a host operating system over which a hypervisor is installed. In our setup, the hypervisor is VirtualBox. It creates virtual environment for multiple VMs one of which here runs a server application that is accessed by the mobile device. Two Ubuntu 16.04 hosts were used as cloudlets and VirtualBox was used to enable virtualization environment to the Virtual Machines.

As the mobile device moves away from the source cloudlet its RTT from it increases. When the RTT goes beyond a threshold RTT, the performance of the application degrades and hence there is a need to connect to another nearby cloudlet. To determine the target cloudlet, source sends a request to all its nearby cloudlet to check its RTT with the mobile device. In return, they send their RTT to source cloudlet, and the minimum of which is chosen as target cloudlet. Each cloudlet has a database which has IP addresses of all its neighbourhood cloudlets and the respective RTT between them.

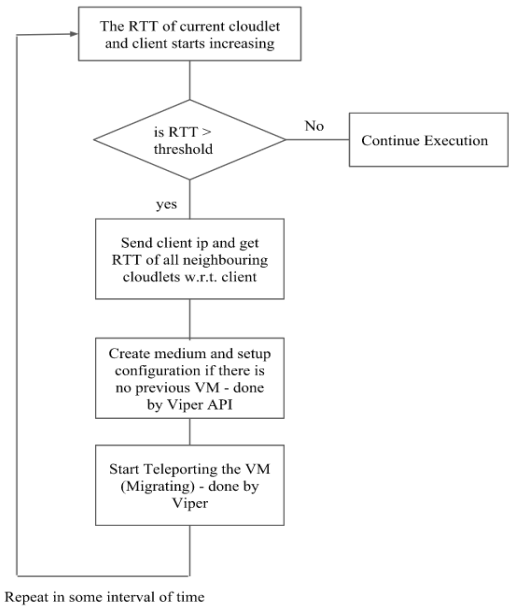

Fig 5: System Flow

A VM is created and configured with respect to the VM in source cloudlet at the destination cloudlet. Teleporter is set on. The folder containing the .vdi file of VM is shared with destination cloudlet using NFS and finally the destination is ready for migration. Migration from source to destination cloud takes place in iteration and hence if the VM is running a resource intensive application, the total migration time will be more, even though the downtime of system is negligible.

7. EXPERIMENTAL SETUP

Host System:

- RAM: 8GB
- Processor: Intel Core™ i5-6200U CPU, 2.30GHz
- OS: Ubuntu 16.04
- OS-Type: 64-Bit
- Disk: 1TB

Guest System (VM):

- RAM: 512/1024/2048 MB
- OS: Ubuntu 14.04
- OS-Type: 64-Bit
- CPUs: 4 to 8
- CAP Execution: 50% to 100%
- Disk: 8GB

VirtualBox Version: 5.1

Python Version: 2.7.2

8. EXPERIMENTAL ANALYSIS

Using the Viper REST API, we have tried Virtual Machine Migration from one host to another. We have tested migration keeping RAM, CPUs, Stress (Load) on CPU and Bandwidth as variables, and calculated migration time and downtime for

these scenarios. These graphs show relation between migration time and downtime with RAM and CPU load.

To study and test VM migration with different CPU load (high computation applications) stress-ng was used to stress test the virtual machine, migrate it and check efficiency of the process. The migration was tested for CPU usage of 25%, 50%, 75% and 100%.

By analysing these graphs, we have concluded some important facts. The following are the various graphs obtained:

8.1 Scenario 1

Downtime is measured by varying the RAM as well as the stress on CPU.

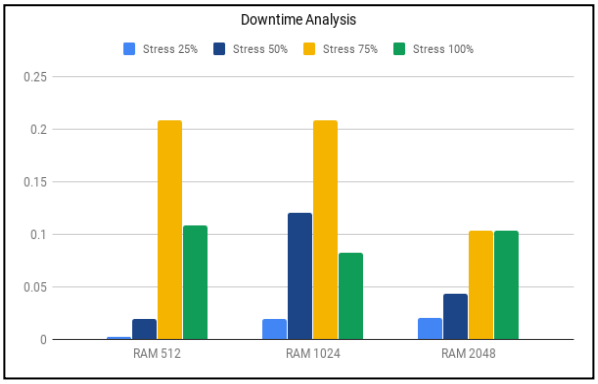

Fig 6: Downtime keeping RAM and Stress Varying

8.2 Scenario 2

Migration time is measured by varying the RAM and CPU.

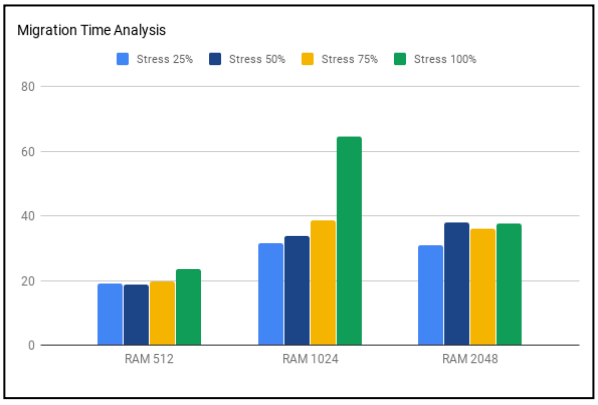

Fig 7: Migration Time keeping Ram and Stress Varying

Table 1. Experiment Results of Downtime Analysis (seconds)

Stress/RAM (MBs)	25%	50%	75%	100%
512	0.0027	0.0192	0.208	0.108
1024	0.0192	0.121	0.209	0.083
2048	0.021	0.04	0.103	0.109

Table 2. Experiment Results of Migration Time Analysis

Stress/RAM (MBs)	25%	50%	75%	100%
512	19.17	31.65	31	19.17
1024	18.758	33.75	37.82	18.758
2048	19.596	38.58	35.99	19.596

Analysis from Scenario 1 and 2:

- As we increase the stress, the migration time increases.
- When the stress is above a certain value and migration is quick, some of the dirty pages are not transferred properly, hence we get a high downtime.
- On the other hand, when the migration is slow, VM's dirty pages are transferred slowly but efficiently, hence the downtime is less in this case.

8.3 Scenario 3

Downtime is measured keeping the bandwidth and Ram constant.

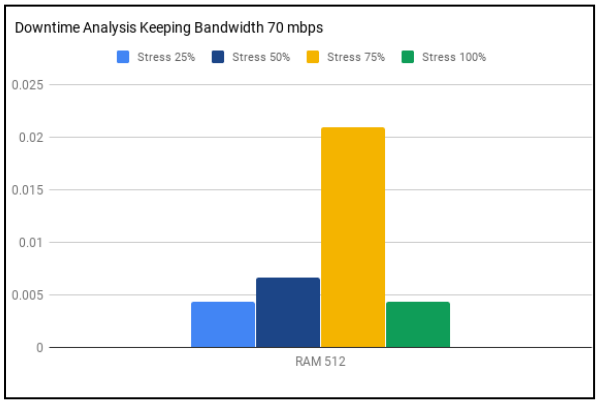

Fig 8: Downtime keeping Bandwidth 70MBPS and RAM 512MB

8.4 Scenario 4

Migration Time is measured keeping the bandwidth and Ram constant.

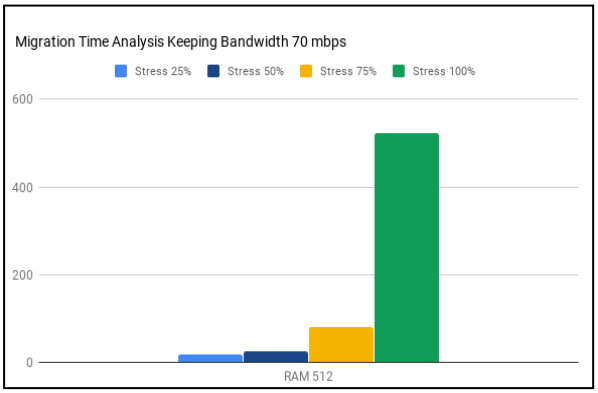

Fig 9: Migration Time keeping Bandwidth 70MBPS and RAM 512MB

Analysis from Scenario 3 & 4:

- As we reduced the bandwidth, the process of migration gets affected.

- The migration time increases, while it does not affect the downtime much.

8.5 Scenario 5

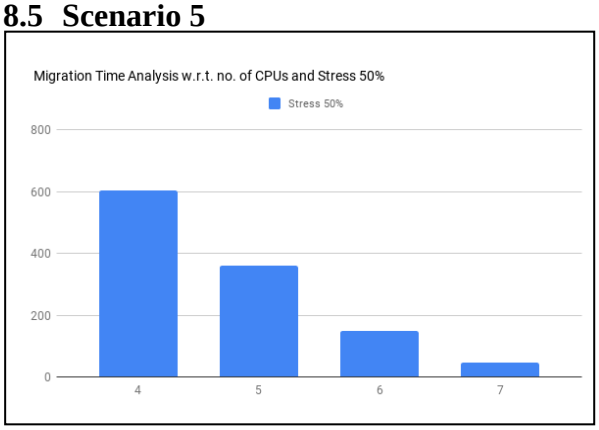

Fig 10: Migration Time keeping Bandwidth 100MBPS, RAM 2048MB, Stress 50% and Varying Number of CPUs

Analysis from Scenario 5:

More the CPUs assigned, faster the dirty pages will get transferred, hence the migration decreases gradually when we assign more CPUs to the VM.

9. CONCLUSION

We discussed the newly emerging paradigms cloudlets and fog computing. The comparison between the cloud and cloudlets gave some light to the development and use of cloudlet to the full potential. Mobile devices and sensors do not have much storage capacity, battery life, and processing power. Using cloudlets would be extremely beneficial and boost to mobile cloud computing and the Internet of Things (IoT). The role of live virtual machine migration in Mobile Edge Computing is very crucial. This concept is just an extension of VM migration in Cloudlets.

The ability to more seamlessly integrate data center with mobile devices considering network connectivity, storage, and computational capability still remains an important challenge in cloud computing. Due to Live Migration, as a user moves across multiple locations, its session while using real-time applications like streaming or gaming can be handed over to multiple cloudlets, thereby maintaining a consistent session for the user.

For these situations, we have developed an API that can manage all the cloudlets, thereby providing uninterrupted use of real-time applications to the user. This API is useful for organizations that have their own cloudlets. The performance of the proposed API for VirtualBox is studied and tested by altering various parameters and each result is concluded.

10. FUTURE WORK

10.1 Multipath TCP

Sometimes, the VM has to migrate to a totally different network. During this process, the IP address of VM changes. This change causes the client device to re-establish TCP connection with VM at the target site. Hence, there occurs a delay in this process. To mitigate this delay, we propose to adopt a mechanism called MultiPath TCP (MPTCP). It allows us to communicate with several IP address/interfaces at a time. Using this technique, the connection between the VM and client will not break even if the IP address of VM changes. Hence, our main future goal is to make Viper support MPTCP. As the IP address of the VM changes, the API should be notified and the corresponding configuration should be made. There might be a need to change the structure of current configuration as right now we are configuring a server only for a single IP address. Moreover, the database regarding neighbouring cloudlets data should be also modified. We currently use IP address as the identifier, but given that IP address will change, the identifier should be a unique key.

10.2 Scalability

We have developed and tested this API at a rudimentary level with 4-5 cloudlets. This API needs to be scaled to handle around 100 cloudlets. Tasks like replication and backup also should be given a good thought.

10.3 Security

There will be a lot of user and business data flowing via this API. Hence, data protection should be ensured thoroughly. One solution is that the data can be encrypted. But, when working with real-time applications, encryption and decryption can take considerable amount of time.

11. REFERENCES

[1] Gursharan Singh, Pooja Gupta, "A Review on Migration Techniques and Challenges in Live Virtual Machine Migration," 5th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) September 7-9 2016.

[2] Fikirte Teka et al "Nearby live virtual machine migration using cloudlets and multipath TCP," Springer Journal of Cloud Computing: Advances, Systems 2016.

[3] Anita Choudhary, Mahesh Chandra Govil et al, "A critical survey of live virtual machine migration techniques," Springer Journal of Cloud Computing: Advances, Systems and Applications 23rd June 2017.

[4] Senthil Nathan et al, "Towards a Comprehensive Performance Model of Virtual Machine Live Migration," Proceedings of the Sixth ACM Symposium on Cloud Computing Pages 288-301

[5] Mattias Forsman, Andreas Glad, Lars Lundberg, Dragos Ilie, "Algorithms for Automated Live Migration of Virtual Machines", The Journal of Systems & Software (2014)

[6] Gang Sun, Dan Liao, Vishal Anand, Dongcheng Zhao, Hongfang Yu, "A new technique for efficient live migration of multiple virtual machines", Future Generation Computer Systems (2015)

[7] Fei Zhang, Guangming Liu, Xiaoming Fu, "A Survey on Virtual Machine Migration: Challenges, Techniques and Open Issues",IEEE Communications Surveys & Tutorials (Volume: PP, Issue: 99)

[8] Latesh Kumar K. J., "Implementing Network File System Protocol for Highly Available Clustered Application on Network Attached Storage", IEEE 5th International Conference on Computational Intelligence and Communication Networks (2013)

[9] Manikandan Selvaganesan, Mohamed Ashiq Liazudeen, "An Insight about GlusterFS and its Enforcement Techniques", International Conference on Cloud Computing Research and Innovations (2016)

[10] Ali Khajeh-Hosseini, David Greenwood, Ian Sommerville, "Cloud Migration: A Case Study of

Migrating and Enterprise IT System to IaaS", IEEE 3rd International Conference on Cloud Computing (2010)

[11] Raghavendra Achar, P. Santhithilagam, Nihal Soans, P. V. Vikyath, Sathvik Rao, Vijeth A. M., "Load Balancing in Cloud Based on Live Migration of Virtual Machines", Annual IEEE India Conference (2013)

[12] Xiang Sun, Nirwan Ansari, "EdgeIoT: Mobile Edge of Computing for the Internet of Things", IEEE Communication Magazine (Volume: 54, Issue:12, December 2016)

[13] Rabiatul Addawiyah Mat Razali, Ruhani Ab Rahman, Norliza Zaini, Mustaffa Samad, "Virtual Machine Migration Implementation in Load Balancing for Cloud Computing", IEEE 5th International Conference on Intelligent and Advanced Systems (2014)