Live VM Migration: An Algorithmic Implementation Approach

Santosh Kumar Majhi
Dept. of Computer Sc & Engg
VSS University of Technology
Burla, India

Suresh Kumar Srichandan
Dept. of Computer Sc & Engg
VSS University of Technology
Burla, India

Sunil Kumar Dhal
Faculty of Research and Doctoral Studies
Sri Sri University, Cuttack, India

ABSTRACT
Recent days the IT industry has shown advancements in the field of cloud computing and virtualization technologies. Virtual machine (VM) is a instance of a computational device, an outcome of virtualization and the heart of cloud computing. Whether it a low end application or a resource intensive software all reside over a VM. It is the VM with which the end user can have access to cloud resources and leverage form them. When talking about the horizontal advancements of cloud the first thing that comes to mind is the accessibility and mobility of the virtual machines. Live VM migration is a step in this direction that allows VMs to be migrated from one server to another without being powered off. Live VM migration has many benefits to it as it helps in effective load balancing and making the cloud fault tolerant. In this paper, we present a novel approach for live VM migration using the concepts of mobile agents. We propose an algorithm for live VM migration for migrating applications among VMs along with migrating virtual machines among various servers. The algorithm is implemented using the Openstack. Finally, we conclude by depicting the working of our algorithm and display the simulated results.

General Terms
Cloud Computing

Keywords
Live VM migration; cloud computing; Virtual Machine; Openstack

1. INTRODUCTION
Cloud computing can be considered as a business model which has introduced a paradigm shift in the industry of Information Technology. As per NIST definition, “cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”[1]. With fast development and exponential growth, it has become one of the key features for every business and is presently used in most of the fields. Cloud computing is still evolving and research is going on to make it a computing model which will be able to work efficiently with the futuristic market.

Nowadays, cloud computing has become an inseparable part of IT industry. Features[2] like scalability, pay per use, platform independency, anywhere available service, elasticity etc. has made it a top choice for business providers. Provisioning, management and scheduling of resources have become very convenient since the outburst of cloud. There are techniques and policies which are being implemented for better handling and automation of compute resources. There are numerous resource management and allocation strategies, one of which is migration of virtual machines. The process of transferring or migrating one virtual machine from one host to another is known as VM migration. VM migration is a resource provisioning technique that is adopted by cloud computing environments in order to prevent Denial of Service (DoS) to an end user. There may be different scenarios when VM migration can take place based upon the need of the cloud provider. It can either be for load balancing or fault tolerance or for maintenance of servers, VM migration finds its application everywhere in cloud. The techniques such as VM migration aim to utilize compute resources to the fullest. Under VM migration[3], a VM is suspended and migrated to another source where its execution is resumed. This kind of a migration process is termed as cold migration. The shutting down and then restarting the VM resulted in low performance. Further advancements where made to this technique and the outcome was to migrate the VMs without shutting them down. This process of migrating VM over different servers without suspension is known as live VM migration.

There are various algorithms and strategies through which live VM migration can be implemented on a real time basis. The aim of this paper is to present an algorithm through which live VM migration can be achieved. The algorithm which we propose comprises of mobile agents, which are simple freely floating computational units. The work of the mobile agents is to gather information regarding migration of a VM or a cloudlet. Whenever there is a condition for migration it is the mobile agent who is informed first. The work of the mobile agents in our algorithm is to look for an appropriate VM or a Host where a cloudlet or a VM can be migrated. The motive of the algorithm is to minimize the calculations pertaining to the migration of a VM and to maximize the performance of migration so that downtime is minimal[4]. The algorithm is deployed and analyzed on a cloud simulation toolkit known as Openstack and the corresponding results are evaluated and represented in the form of graphs.

The rest of the paper is categorized as follows: In Section II elucidates the background of the work. Section III talks about the related work in the field of live VM migration. We explain our proposed algorithm in Section IV and its simulation in detail in Section V. Finally, Section VI provides a brief summary and concludes the paper.

2. BACKGROUND
The aim of this section is to present background information on the some topics which form a basis for the other viewpoints which are discussed in this paper.

Mobile Agents: Mobile agents are software components which travel autonomously across different machines with
very less overhead. These are very small computational entities that majorly focus on moving randomly around hosts so as to collect information which can be used to carry out other processes. They use Remote Procedure Calls to facilitate one to one transition from one host to another. In case of deploying multiple mobile agents at different levels, one needs to establish a shared system in order to carry out a single process in a parallel. The main characteristics of mobile agents is their mobility, low network latency, robustness, light weight, parallel processing, easy and flexible management, dynamic adaptation and fault-tolerant capabilities [5]; amongst all mobility is of prime importance. Mobile agents find their applications in many domains and recently, IBM Japan deployed a system based on mobile agents written in Java named as Aglets [6].

Live VM Migration: Live migration [8] "refers to the process of moving a running virtual machine or application between different physical machines without disconnecting the client or application. Memory, storage, and network connectivity of the virtual machine are transferred from the original guest machine to the destination". While considering live VM migration, four key phases are present which are [7]:

a) CPU state: In case of live VM migration CPU state comprises of the current and future CPU utilization of a VM. It is an essential content to be migrated as it is the CPU state which governs the uninterrupted execution of applications on the destination VM. This information is of high importance for smooth running of applications. It can be taken as a minimum bound to map the downtime of migration process.

b) Memory content: It comprises the memory state of all active processes running in the VM which is to be migrated. The information about the VM and the OS running in it are part of the memory content. This information is very large in size as compared to the CPU state and thus certain techniques need to be implemented in order to minimize its the migration time.

c) Storage content: Storage content comprises of the VM disk image. Storage content of a VM is extremely large in size and its migration results in consumption of large amount of resources. Generally, migration of storage content is considered as an optional part as it is not mandatory to migrate the disk components when both the source and destination hosts are connected through a NAS.

Pre-copy live VM migration: This is a technique is used for migration of memory content of VM [8]. The three phases under pre-copy migration are:

a) Identifying Dirty pages: Before starting migration, all the memory pages of the VM are set to read only mode so as to detect the modified pages. Once the migration is in process then the memory pages can undergo modifications by the applications running on the VM. These modified pages are marked and stored as a bitmap i.e. dirty bitmap. These dirty pages are then copied again to the destination server.

b) Iterative phase: In this phase, dirty bitmap is observed and the dirty or modified pages are prepared for migration again. The dirty pages are again set to read-only and they are migrated to the destination host where they replace their earlier version of copy.

c) Stop and copy: The VM executing at the source end is suspended and iteration takes place identifying all the dirty pages followed by their migration to destination server. The VM is then resumed on the destination server marking the success of migration process.

The performance of pre-copy migration dips when the amount of dirty pages is very large. Subsequent change in the pages results in a large number of iterations and thus overall performance gets degraded.

Post-copy live VM migration: Live VM migration using post-copy technique begins by suspension of VM at source end. All the necessary pages and memory content required for the VM to start its execution at destination end are migrated. VM is resumed at destination server. There may be a chance when some pages are not migrated which indeed results in page faults. [9]. These page faults are serviced by source server which sends the corresponding page to destination server as soon as possible. The lesser number of page faults, more is the performance and vice-versa. In post-copy approach, a page is sent only once to the destination side whereas in pre-copy approach, a page can be sent over multiple iterations i.e. whenever it is dirtied, it is resent. Pre-copy keeps the updated version of VM whereas in post-copy approach, VM state is distributed and can thus result in ambiguity.

Openstack: Openstack is a cloud OS used for cloud computing environments. Some novel features of Openstack are [10]:

a) Provides support for large scale as well as small scale modelling and simulation of cloud computing environment which will include components like data centres, virtual machines, resources, hardware etc. and their provisioning.

b) Free-standing and complete platform for deploying the scheduling algorithms, resource allocation, and data centre management etc.

c) Presence of virtualization engine aids in creating multiple independent virtualized services on Openstack.

d) Can switch the functioning between time shared and space shared policies.

e) Supports modelling in viewpoint of system and behaviours for components like data centres, virtual machines and other resource management techniques.

f) A single cloud or federation of cloud both can be deployed on Openstack.

3. RELATED WORK

Since the past couple of years, researchers have formulated various algorithms and techniques in order to undergo migration of virtual machines. Continuous work is still going on so as to device more optimized and efficient algorithms where in VM can be migrated with minimum downtime. In this section, we have presented some of the research works pertaining to this area.

Data Deduplication [11] is a technique for live VM migration that avoids migration of massive chunks of data thereby reducing migration time. It works on the principle of migrating only selected memory contents that have been altered at the source server. The migration process includes only those portions of the VM that have been modified at the source end. The two major components working behind data deduplication are DBT and Diff format. DBT stands for Dirty Block Tracking and its work is to record all the operations that cause changes in the VM disk image while Diff format is
used for storing the recorded results. DBT keeps track of every modified disk page and marks it as a dirty page. Only the pages identified by the DBT undergo storage migration and rest are left behind. Data deduplication is beneficial for those VMs that undergo multiple migrations and thus leading to multiple destination servers. It is one of the most efficient techniques for live VM migration as it reduces the migration time by a factor of 10.

Shrinker[12] is a system for live migration of virtual machines, allowing migration of VM clusters between data centers connected through a network. Shrinker reduces the data to be migrated by incorporating the concepts of data deduplication and cryptographic hash functions. It works on the principles of distributed content based addressing and thereby allowing chunks of VMs to be migrated along various servers in multiple data centers. Shrinker is different from traditional ways of live VM migration as it allows hypervisors residing on source and destination servers to interact with one another during migration. It makes this possible by implementing interconnecting services at both ends. During migration VMs are divided into chunks of data blocks so as to ease the migration of data between servers. These data blocks are then mapped by the cryptographic hash function and assigned unique hash values accordingly. Shrinker has a coordination service running at the source end while an indexing service at the destination. The work of the coordination service is to receive hash values and accordingly migrate data from the source server. On the other side the indexing service registers every data block according to its hash value which are then assigned to a specific destination server. The destination server coordinates with the indexing service in order to assemble various data blocks into a VM.

Inter cloud live VM migration[13] is a new way of looking towards VM migration. It allows VMs to be migrated not just between data centers of the same cloud but also among servers residing on different clouds. The driving force behind inter cloud live VM migration is to minimize the workload on a particular cloud and reduce its network congestion. It works on the concept of creating snapshots of the VM which is to be migrated. The snapshot is then migrated to the destination cloud where the hypervisor is directed to create a new VM having the same configuration as mentioned in the snapshot. Soon after the destination VM is up and running the source cloud redirects the incoming traffic of its VM to the destination VM. Techniques such as effective Fault tolerance incurs the benefits of inter cloud live VM migration.

Work on opportunistic replay [14] aims at reducing the amount of data being migrated in case of low bandwidth environments. This approach keeps a record of all kinds of user events taking place during VM execution. This information is then transferred and put into effect on an identical fabricated VM so as to produce nearly the same state as the source VM. The changes that have been made after the replay are also transferred and applied, resulting in an identical surrogate VM.

EnaCloud[15] is an energy saving application that facilitates application scheduling along with live VM migration. The purpose of this application is to reduce down the number of active VMs and the number of migrations that take place in order to cut down the energy consumption. The architecture of EnaCloud comprises of two nodes namely, storage and computing nodes. The storage node is used for storing data files while the computation node acts like a host for a number of VMs. The server nodes having VM running on the are termed as closed boxes whereas idle server nodes are known as open boxes. The main aim of EnaCloud is to reduce the number of open boxes by incurring the concepts of workload resizing. Every time a new workload (VM) arises the application maps the workload to a resource node depending upon the resource requirements of the workload. This mapping is done by the means of live migration so as to minimize the number of closed boxes.

4. PROPOSED ALGORITHM
VM migration can either take place from step 1 or from step 10.

1. If (RU factor of VM >= threshold value)
2. Notify MA1
3. MA1 searches for appropriate VM
4. If (search == success)
5. Notify source and destination VM
6. Cloudlet created at the destination VM with same configuration of the source VM
7. Print (“migration successful”)
8. Else (search != success)
9. MA1 will inform the server and goto step 11
10. If (RU factor of server > threshold value)
11. MA2 searches for appropriate server for VM migration
12. If (search == success)
13. Notify source and destination server
14. VM migration takes place in running state
15. New VM created at the destination server with the same configuration of the source VM
16. Cloudlets created of same configuration at the destination VM
17. Source VM deleted
18. Print (“migration successful”)
19. Else (search! = success)
20. Server reached in compromising state
21. Input (“cloudlet id to be stopped or removed”)
22. Respective cloudlet stopped or removed

5. IMPLEMENTATION & RESULT
After successful implementation of the algorithm on OpenStack the following screenshots depict the simulated results.

The graph shows increase in the RU factor of a server with respect to time. The Y axis represents time (minutes) whereas the X axis represents RU factor. Resource utilization of a server starts at time t=2 min and keeps on increasing till it reaches its threshold value RU=5 at t=10 min. At this very instance of time MA2 searches for an appropriate host leading to live VM migration. The resultant of which would cause a decrease in the RU factor i.e. RU=4 at t=13 min.
6. CONCLUSION

Virtual Machine’s provisioning and migration are one of the most important aspects of virtualization when it comes to cloud computing. With the ever increasing load on cloud datacenters it becomes essential to devise new ways of managing this load alongside maintain the Quality of Service (QoS). Live VM migration is one of the best known solutions to this problem as it allows migration of VMs without causing the applications running on them to shut down. In this paper, we lay emphasis on this concept and present our own algorithm for live VM migration. The algorithm makes use of certain aspects of mobile agents which makes it more robust and easy to deploy. We implement and simulate this algorithm on Openstack which is a cloud framework. At the end we have the simulated results which are analyzed and well explained through graphical representations.

7. REFERENCES


