Dynamic Load Balancing of Virtual Machines using QEMU-KVM

Akshay Chandak
Krishnakant Jaju
Department of Computer Engineering and Information Technology,
College of Engineering, Pune.
Maharashtra, India.

Akshay Kanfade
Pushkar Lohiya
Department of Computer Engineering and Information Technology,
College of Engineering, Pune
Maharashtra, India

Amit Joshi
Department of Computer Engineering and Information Technolgy,
College of Engineering, Pune.
Maharashtra, India.

ABSTRACT

Virtualization technologies share the hardware resources among multiple operating systems and maintain isolation between virtual machines. Thus, they are used to optimize resource utilization, minimize job response time and for more efficient use of servers and other resources. For this, it is necessary that the load is evenly distributed over all the hosts in the network. The proposed work emphasizes on the design and implementation of a policy engine to dynamically balance the load over a network, using live migration feature of KVM. The goal is to provide a provisioning monitor that can dynamically make decisions about migration of heavily/lightly loaded virtual machines.

Keywords

Virtualization, Load balancing, QEMU-KVM, Live Migration, Virtual Machine (VM)

1. INTRODUCTION

Virtualization [1] is commonly defined as a technology that introduces a software abstraction layer between the hardware and the operating system and applications running on top of it. Its main advantages include isolation, consolidation and multiplexing of resources. Other benefits of virtualization include saving on power by consolidation of different virtual machines on a single physical machine, migration of virtual machine for load balancing etc. Virtualization provides full control of resource allocation to administrator, resulting in optimum use of resources. More recently, another advantage of virtualization - live migration of virtual machine is increasingly used to better handle workload balancing across physical machines in data center, especially when the available resources in physical machine are not sufficient for VMs.

The load balancing [2] problem determines how to divide work between available machines in a way that achieves performance goals. In static load balancing, the mapping of jobs to resources is not allowed to change after the load balancing has begun. On the other hand, dynamic load balancing will change the mapping of jobs to resources at any point, but there may be a cost associated with the changes. Load balancing is essentially a resource management and a scheduling problem. The operating system on each host performs local scheduling. Local scheduling involves deciding which processes should run at a given moment. Global scheduling, on the other hand, is the process of deciding where a given process should run. Global scheduling may be performed by a central authority or it can be distributed among all the hosts. The goal of the proposed work is to dynamically balance load on the hosts by a central authority, which is also a virtual machine.

Section II gives an overview about Virtualization and live migration and other prerequisites for the setup. Section III explains load balancing concept and phases involved in it. Section IV proposes the algorithm for dynamic load balancing based upon the framework given in Section III. Results are shown in Section V. Finally, some conclusions are drawn in Section VI.

2. VIRTUALIZATION AND LIVE MIGRATION

Core of any virtualization technology is Hypervisor or Virtual Machine Manager (VMM). Hypervisor is a piece of software which allows each virtual machine to access and schedule the task on resources like CPU, disk, memory, network, etc. At the same time hypervisor maintains the isolation between different virtual machines. Virtualization can be classified by the method in which hardware resources are emulated to the guest operating system [1]. They are as follows:

2.1 Full Virtualization

Hypervisor controls the hardware resources and emulates it to guest operating system. In full virtualization, guest does not require any modification. KVM is an example of full virtualization technology.

2.2 Paravirtualization

In paravirtualization, hypervisor controls the hardware resources and provides API to guest operating system to access the hardware. In paravirtualization, guest OS requires modification to access the hardware resources. Xen is an example of paravirtualization technology.

2.3 KVM

Kernel-based Virtual Machine (KVM) [9] project represents the latest open source virtualization technology. KVM is implemented as a loadable kernel module that converts the Linux kernel into a bare metal hypervisor. In the KVM architecture, the virtual machine is implemented as regular Linux process, scheduled by the standard Linux scheduler. In fact each virtual CPU appears as a regular Linux process. This allows KVM to benefit from all the features of the Linux kernel.

2.4 QEMU
Q-Emulator [7] is a generic open source processor emulator and virtualizer. It can run many Operating Systems and programs made for one machine on another. It uses dynamic binary translation to achieve high performance. Binary Translation is an emulation technique in which, instead of emulating the processor, the virtual machine runs directly on the CPU [7].

2.5 Shared Storage using Network File System (NFS)
The hosts in a live migration write to the same VM image file when a VM is moved. Thus, Live migration requires shared storage which is provided by NFS.

2.6 Live Migration
Live Migration [6] of a virtual machine is simply moving the running VM on a physical machine (source host) to another physical machine (target host) without disrupting any active network connections, while the VM is running on the source host, even after the VM is moved to the target host. It is considered live, since the original VM is running, while the migration is in progress. Very small downtime, in the order of milliseconds, is the benefit of doing live migration. We can migrate a guest between an AMD host to an Intel host and back. Naturally, a 64-bit guest can only be migrated to a 64-bit host, but a 32-bit guest can be migrated to 32 or 64 bit host.
The live migration task comes down to the following steps [10]:

2.6.1 Pre-Migration
Select VM to be migrated and destination host where resources required are guaranteed to be present.

2.6.2 Reservation
In this stage, confirmation of necessary resources at destination host is done and a VM container of that size is reserved.

2.6.3 Iterative Pre-Copy
The pages from guests physical address space are sent one at a time to the destination system i.e. the guests memory is copied to the destination. The pages that are copied are made read-only for the guest during the live migration process. In the following iterations, only the dirtied pages are copied.

2.6.4 Stop and Copy
At this stage, VM at source host is suspended and network traffic is redirected to destination host. Also CPU state and any remaining inconsistent memory pages are then transferred.

2.6.5 Commitment
Now, Destination host indicates source host that it has successfully received a consistent VM image.

2.6.6 Activation
The migrated VM on destination host is now activated.

3. LOAD BALANCING
Load balancing is the process of reallocating VMs on another host in the network in order to improve resource and network utilization. Common goals of load balancing include maximizing throughput, minimizing response time, and/or minimizing communication time and avoiding the scenario in network that, some hosts are under-utilized and some over-utilized. The important factors to consider while developing such algorithm are estimation of load, comparison of load, performance of systems, nature of work to be transferred and selection of hosts [2].
Static load balancing is VM placement problem. Here, the host on which VM will be placed is decided before it starts running depending upon the load on the network i.e. host with least system usage runs the VM. Dynamic load balancing reassigns VMs based on system performance at run time using the feature of live migration of QEMU-KVM.

3.1 Goals of Load Balancing
Following are the goals of load balancing as described in [2]:

- To improve the performance substantially
- Fault tolerance in case of system failure
- To maintain the system stability
- To accommodate future modification in the system

3.2 Types of Load Balancing Algorithms
Following are types of load balancing algorithms as described in [2]:

Sender initiated	: Algorithm initiated by Sender
Receiver initiated	: Algorithm initiated by Receiver
Symmetric	: Combination of above two

3.3 Previous Work
The five phases of load balancing as described in [8] are:

- Load Evaluation
- Profitability Determination
- Work Transfer Vector Calculation
- Task Selection
- Task Migration

The Central Scheduler Load Balancing (CSLB) [4] uses a central node that makes all load balancing decisions. It uses the above five phases for VM migration instead of process migration as proposed in [10].

Policy Engine [3] is heart of load balancing algorithm. It decides when to migrate virtual machines between hosts and runs as a normal virtual machine. The aim behind this is, it can move itself to a different host like any other virtual machine, depending on the load.(Fig 1)

4. PROPOSED ALGORITHM
The proposed algorithm is modified version of Central Scheduler Load Balancing (CSLB) algorithm [4]. The algorithm uses the five phases for load balancing as described above:

4.1 Load Evaluation
As stated, the algorithm is a central scheduling algorithm, where all the hosts send load information to the policy engine: which is responsible for load balancing decisions, after a predefined time interval which can be changed as per the requirements.

The CPU utilization is divided into three bands: Lightly loaded, moderately loaded and heavily loaded based on the threshold value as shown (Fig 2).

The threshold is the average of CPU usage of all hosts. The moderately loaded band is created by adding and subtracting a value, which is the difference between the average and the mean of the maximum and minimum of the CPU usage of the hosts or of 20 percent width, across the threshold, whichever is maximum. All the hosts fall in at least one of the bands.

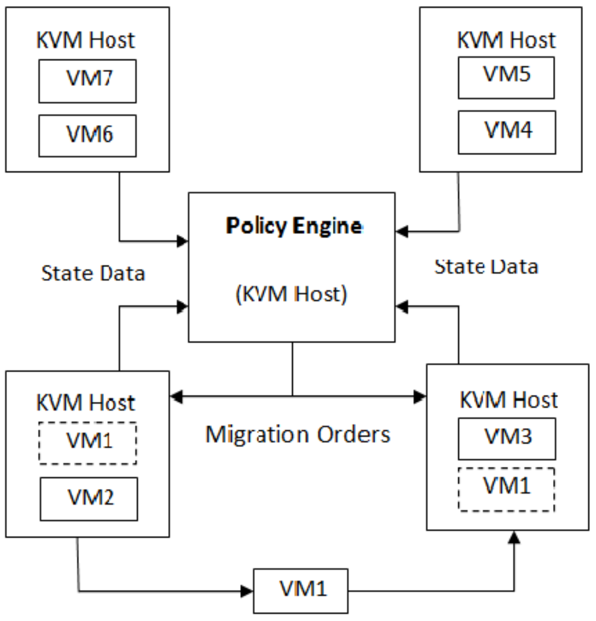

Fig 1: Relationship between policy engine and the KVM hosts.

The systems would be in an ideal state if all the hosts lie in the moderately loaded band.

Calculation of moderate band:

threshold = $(\alpha_1+\alpha_2+\alpha_3+...+\alpha_n)/n$

mean = $(\alpha_{min}+\alpha_{max}) / 2$

diff = |*threshold* - *mean*|

if diff > 10

  moderately loaded band = threshold ± diff

else

  moderately loaded band = threshold ± 10

where,

α_i= *CPU usage of* i^{th} *Host over a defined time in percentage and i = 1 to n*

4.2 Profitability Determination

Migration of virtual machines from one host to another will be considered profitable if there exists one virtual machine in the heavily loaded band and one in lightly loaded band.

4.3 Work Transfer Vector Calculation

For hosts that are not in the moderately loaded band, load balancing policies are applied so that they end up in a moderately loaded state. This is done by transferring a VM from heavily loaded band to a lightly loaded band such that both the systems try to achieve a moderately loaded band. The value is calculated by taking the difference between the threshold and the current CPU utilization of the lightly loaded host.

4.4 VM Selection

VM Selection can be estimated using the virtual machine usage on a host. The virtual machine that has a usage closer to the amount calculated in phase 3 (i.e. Work Transfer Vector Calculation) should be migrated.

4.5 VM Migration

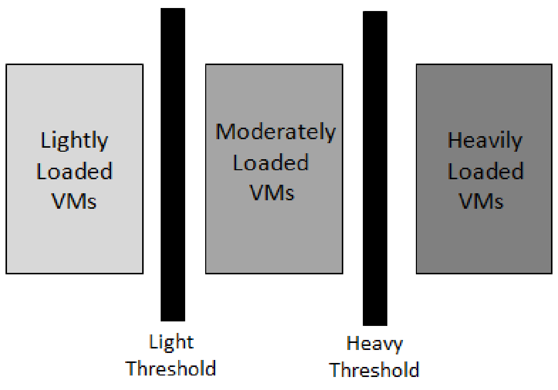

Fig 2: Each host machine is assigned to one of three groups based on CPU utilization and two thresholds. A darker color indicates that the host is more heavily loaded.

Here QEMU-KVM's live migration feature is used. It is done using the virsh command which is a libvirt API (a toolkit to interact with Virtualization capabilities of OS).

The relation between five phases of the algorithm is explained by the following flowchart (Fig 3):

5. PERFORMANCE AND RESULTS ANALYSIS

In order to assess the performance of the algorithm, a prototype system of VM management was developed. Virtual platform: KVM and storage system: NFS was used. A physical machine was chosen as the host machine. QEMU-KVM and Virtual Machine Manager were installed to manage and schedule VM; and its operating system was Fedora 15, CPU: Intel Core i5-2400 3.10 GHz * 4, and Memory: 3 GB. Three client machines of same configuration as above were chosen. libvirt, qemu-kvm, virt-manager, nfs server packages were installed.

In this scenario the CPU usage sent to policy engine is the average of CPU usage over last three minutes of respective hosts. The policy engine calculates thresholds every five minutes and takes the decision for load balancing i.e. whether to migrate some VM or not.

The experimental results are as follows (CPU USAGE in percentage). Table 1,2,3 and 4 depicts the hosts and their corresponding CPU usage after three iterations of the algorithm. Followed by Fig. 4 which depicts the variation in CPU usage of hosts over period of 25 minutes. The usage of CPU by individual VMs was approximately constant over 25 minutes and each VM was running on single core.

By using the formulae given in Load Evaluation, lower and upper limit of moderate band are calculated. Hence Host 1 is in moderately loaded band, Host 2 is in lightly loaded band and Host 3 in heavily loaded band and A2 is migrated from Host 3 to Host 2. (Table 2)

By using the formulae given in Load Evaluation, lower and upper limit of moderate band are calculated. Hence Host 1 is in moderately loaded band, Host 3 is in lightly loaded band and Host 2 is in heavily loaded band and K2 is migrated from Host 2 to Host 3. (Table 3)

By using the formulae given in Load Evaluation, lower and upper limit of moderate band are calculated. Hence Host 1 is in moderately loaded band, Host 3 is in lightly loaded band and Host 2 is in heavily loaded band and K3 is migrated from Host 2 to Host 3. (Table 4)

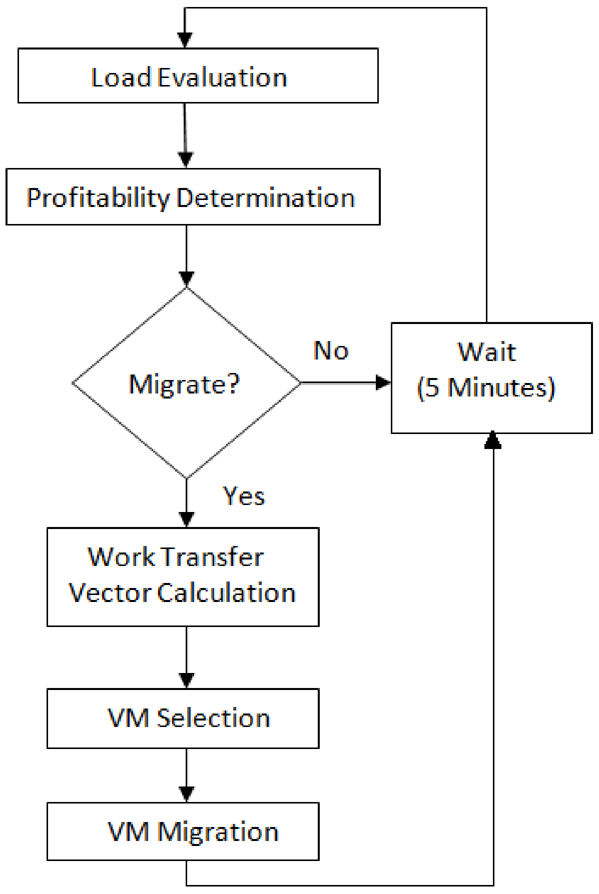

Fig 3: Flowchart of five phases of algorithm

Table 1. State of VMs on Hosts

Hosts	VM Name	VM's CPU Usage on respective core	Host's CPU Usage
Host 1	S1 S2 S3	46.50 9.60 3.00	42.4
Host 2	K1 K2 K3	48.50 8.70 23.80	19.07
Host 3	A1 A2 A3	93.50 24.80 2.90	62.18

By using the formulae given in Load Evaluation, lower and upper limit of moderate band are calculated. Hence all hosts are in moderately loaded band.

Table 2. State of VMs on Hosts

Hosts	VM Name	VM's CPU Usage on respective core	Host's CPU Usage
Host 1	S1 S2 S3	49.40 10.10 1.00	40.99
Host 2	K1 K2 K3 A2	49.60 7.00 24.50 23.40	53.33
Host 3	A1 A3	94.50 1.90	29

Table 3. State of VMs on Hosts

Hosts	VM Name	VM's CPU Usage on respective core	Host's CPU Usage
Host 1	S1 S2 S3	51.40 10.40 1.00	42.58
Host 2	K1 K3 A2	46.10 25.10 51.90	50.59
Host 3	A1 A3 K2	95.10 1.80 2.10	27.31

Table 4. State of VMs on Hosts

Hosts	VM Name	VM's CPU Usage on respective core	Host's CPU Usage
Host 1	S1 S2 S3	52.20 10.60 1.00	41.17
Host 2	K1 A2	41.80 61.90	35.77
Host 3	A1 A3 K2 K3	95.60 1.70 1.00 24.70	38.78

6. CONCLUSIONS AND FUTURE WORK

The work has proposed a policy engine to dynamically balance the load over the network. Originally the network was imbalanced. There were hosts in heavily as well as lightly loaded bands. After some iterations (three in the above scenario) of the load balancing algorithm, all the hosts were balanced i.e. all the hosts were in the moderately loaded band (Fig. 4).

Cloud Computing is a vast area and load balancing plays a very important role in case of Cloud. The work has focused on CPU usage as load parameter that is applied to the setup, but there are still other parameters and approaches that can be applied to balance the load. The performance of the given algorithm can be increased by varying different parameters like memory usage, disk I/O, network load.

Further dynamic load balancing can be improved and the Live Migration algorithm implemented in QEMU-KVM can be optimized, so that migration time will be reduced and performance will also be improved.

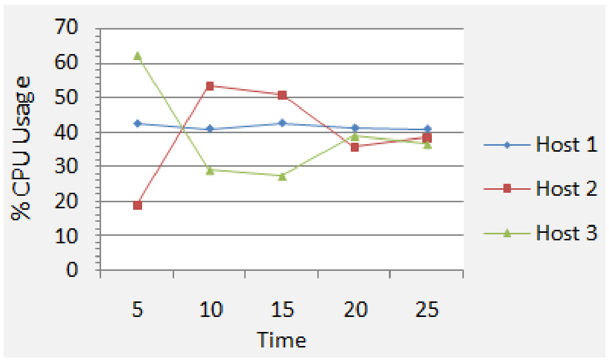

Fig 4. Variation in CPU Usage of Hosts

7. REFERENCES

[1] Jyotiprakash Sahoo, Subasish Mohapatra, Radha Lath, Virtualization: A Survey On Concepts, Taxonomy And Associated Security Issues, Second International Conference on Computer and Network Technology, 2010.

[2] Ali M. Alakeel, A Guide to Dynamic Load Balancing in Distributed Computer Systems, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.

[3] Terry C. Wilcox Jr, Dynamic Load Balancing Of Virtual Machines Hosted On Xen, Department of Computer Science, Brigham Young University, M.S. Thesis, April 2009.

[4] Youran Lan, Ting Yu, A Dynamic Central Scheduler Load Balancing Mechanism, Computers and Communications, pp 734-740, May 1995.

[5] Yi Zhao, Wenlong Huang, Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud, Fifth International Joint Conference on INC, IMS and IDC, 2009.

[6] F. Ma, F. Liu, Z. Liu, Live Virtual Machine Migration Based on Improved Pre-copy Approach, Proc. Software Engineering and Service Sciences, pp. 230-233, 2010.

[7] Geoffroy Vallee, Thomas Naughton, Christian Engelmann, Stephen L Scott, Hong Ong, System-level Virtualization For High Performance Computing, 2008.

[8] Jerrell Watts, Stephen Taylor, A Practical Approach to Dynamic Load Balancing, IEEE Transactions on Parallel and Distributed Systems, Feb 1998.

[9] KVM Kernel Based Virtual Machine Red Hat, Inc. 2009.

[10] Christopher Clark et. al., Live Migration of Virtual Machines, 2nd Symposium on Networked Systems Design & Implementation, NSDI’05