CPU and Memory Utilization by improving performance in network by Live Migration technology

Igli TAFA[1], Elinda KAJO[2], Hakik PACI[3], Elma Zanaj[4], Aleksandër Xhuvani[5]

**Polytechnic University of Tirana, Information Technology Faculty,Computer Engineering Department
Tiranë, Albania.**

Abstract
The aim of this paper is to improve the performance of network communication between some Virtual Machines in LAN by modifying a script in Xen. Also in this paper we have tested the utilization of CPU and Memory during the live migration phase. After these tests we have concluded that there is no dependency between Memory of Virtual Machines and CPU Consumed. These experiments are performed in Xen Hypervisor, because it offers para-virtualization approach which support more flexibility for all Guest Operating Systems.
Keywords: *Virtual machines, Network Performance, Xen, Para-Virtualization, Guest Operating System.*

1. Introduction

Virtualization is a global technique which simultaneously can execute more than one machine independently. A strong tool used by this technique is Live Migration. It means transferring application, memory pages, CPU-status, network-status etc from one machine to others. All these possibilities are offered by the Hypervisor, which is built in the Bare Hardware or above the Host-Operating System. There are a lot of hypervisors such as, VM-Ware Workstation, ESX-Server, Virtual-Box etc. Most of them are close source or in some cases, they do not offer diversity in resource management. One of the most popular hypervisors is Xen. In this paper all the tests are performed with this hypervisor. The reasons of using Xen Hypervisor are :

- It is based on Para-Virtualization approach which offers more flexibility in Resource Management.
- It is an open-source system which gives the possibility to introduce our additional tools in different kinds of experiments such as: load balancing with memory ballooning approach [1] , CPU-performance of activities, Memory Compression [2] etc.

Xen operates above the bare hardware (Fig. 1). Also Host Operating System is built on the bare hardware, and it is called Dom0. Above Xen are located the Guest Operating Systems, or Virtual Machines. These machines are aware of Physical Resources connected with the hardware machine. They call DomU and can communicate with each-other as if been in the physical machines connected by a network. So in these machines we should configure respectively their IP address.
As it looks from the fig.1 the Hypervisor is located above the bare hardware. In order to modify the hypervisor we should compile the Kernel of Host Operating System. Above it there are the virtual machines and on the top there are the applications.
One of the main problems in Xen is the overhead. It is caused from live migration techniques. Xen uses a memory sharing mechanism, called the *grant* mechanism. This mechanism is used to share I/O buffers between Guest Operating Systems memory resident on Hypervisor. In [3] is presented an improvement mechanism which reduces the number of grant issue and provides a unified interface for memory sharing using IOMMU hardware between guest domains and I/O devices.

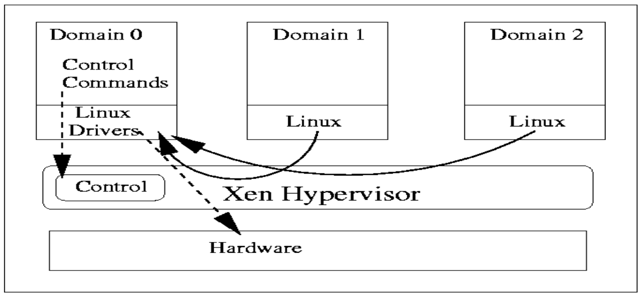

Fig.1 This is the architecture of Xen Hypervisor

Xen can support up to 100 virtual machines, but practically this number reduces dramatically. To achieve a good performance Xen can support up to 16 virtual machines. In Xen every machine has it`s own dedicated memory. For instance if we have 4 GB RAM and we have built 3 virtual machines above Xen, each machine should take just 1 GB of memory from RAM, because in generally 1 GB RAM is dedicated for the Host Machine.

Inside a P hysical Machine we can built a Cluster system. It means that a large number of virtual machines (more than 2) can communicate with each other in a synchronization form. The cluster system means that one of the selected computers from a group is *master computer* and it is responsible to manageable a lot of activities between *slave computers*. This approach gives a better performance, notably when we are introduced to High Computing Applications such as large matrices multiplication. One tool which reduces communication speed is network interfaces and protocol communications. If the slave computers are communicating with 10/100 Mb ethernet the total performance will degrade. But if we use gigabit ethernet communication speed the time will reduce so the performance will increase.

Most of the communication protocols between applications in different hosts are TCP and UDP. As we know, UDP is more flexible and faster than TCP, but these analyses don't take in consideration the use of Cluster system in Xen. In this way the master-slave system, introduced as the Virtual Machines and Cluster system above Xen are called Virtual Cluster. At this moment the main problem is Single Point of Failure Server. If the physical machine goes down i.e from breakdown AC or System Bugs all the Virtual Cluster will fail.

In this paper we have to combination the physical and virtual clusters. We want to test the CPU performance in load balancing during the generation of requests from the source machine to the target one, Physical and Virtual Memory Utilization in case of warning failure and all network activities. The failure may happen accidentally i.e from AC breakdown, or natural reasons such as: fire, earthquake, tsunami etc, or it can be forecasted from us. We can analyze only the second case.

Cluster system should give high availability features such as: Backup Storages (HD, DVD, Tape, RAID technique, SAN etc), Backup of Power Resources (UPS, Inverter, Second AC line connection), Backup Data Broadcast Lines etc.

This paper is organized as follows. In the second section is introduced the Live Migration technique. In the third section we have presented the Experimental Phase. In the forth section are given the conclusions and future works, and in the fifth section are the references.

2. Live Migration technique

As we wrote above this technique is a strong tool for transferring application, memory pages, CPU-status, network-status etc from one machine to the others. The live migration can be realized in two ways: Pre-copy and Post-copy approach [4]. In [4] post-copy iterative approach has a better performance then pre-copy approach. This happen because initially it transfers just CPU status between machines, then Memory free pages and Dirty pages. In [2], it is used a "zero-aware" algorithm approach which improves the total performance of live-migration. A necessary condition for live migration approach between virtual machines is accessing of them on the shared storages. System images are built in the storages. If a virtual machine fails, automatically a copy of this machine is migrated to other virtual machines. This migration is managed by the Hypervisor layer.

Cluster system is a physical computer's system which is connected with each other in LAN (Local Area Network). In figure (2) we present a cluster computer system. There are three physical hosts with some virtual machines inside (VM1,…VM5). Based on this figure if any physical machine fails immediately, all the information of this machine is transferred across the network to other physical hosts. As we wrote above, this approach is valid in the controller failure approach, but if the fail is suddenly (accindental reasons) nothing is transferred.

In figure 2 each machine uses a portion of Data – Center, SAN (Storage Area Network), which is called virtual data center and can be used by every virtual machine. Some applications in the data center can be moved from one portion of virtual SAN to another one.

Another approach in live migration is the management of CPU performance. If one virtual machine is performing a lot of tasks, hypervisor moves some of these tasks from the specified machine to another machine with a lower load.

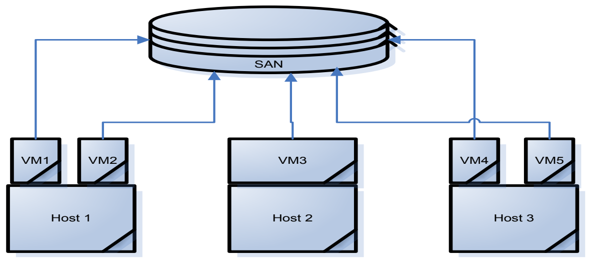

Fig. 2 Five virtual machines and three physical hosts in the cluster system connected with SAN

Another key topic in Live Migration approach is the detection of hotspots. In [5] is implemented a sandpiper architecture which detects hotspot by gathering information from two tools, *black box* and *gray box*.

Sandpiper implements a hotspot detection algorithm that determines when to migrate virtual machines. Also it determines where to migrate and how much

allocation should be used for the memory machine after this migration.
The hotspot detection component employs a monitoring and profiling engine that gathers usage statistics on various virtual and physical servers and constructs profiles of resource usage.

3. Experimental phase

We will perform some experiments based on two objectives:

3.1 Check the CPU performance
3.2 Check the network performance

In 3.1 we will check the CPU performance:

- Between two virtual machines on a physical host
- Between two virtual machines in different hosts in the same LAN
- Activities in which Virtual Machine`s Memory increases and decreases

In 3.2 we will check the Network performance in Cluster Physical System machines:

- CPU Performance between two virtual machines in different physical hosts.
- Additive time causes from Warning Failure approach in a specific physical host.

The parameters for all experiments are as follows:

1. Architecture x86 32 bit machine.
2. Computer model HP Dual Core with HT
3. RAM 4 GB.
4. Dom0 Ubuntu 10.04 Server.
5. DomU1 Ubuntu 10.10 Desktop.
6. DomU2 Windows XP.
7. Xen Hypervisor version 4.0.1.
8. Apache installed in DomU1 version 2.2.16 (LAMP package).
9. Apache installed in DomU2 (WAMP Server 2.1a).
10. Heartbeat benchmark installed in Dom0.
11. Httperf benchmark installed in DomU1.

3.1 Check the CPU performance

3.1.1 CPU performance between two virtual machines on a physical host

The evaluation of CPU performance between two virtual machines in a physical host starts with the installation of httperf benchmark in DomU1 and WAMP Server 2.1a in DomU2. DomU1 sends 1000 requests in second in the Apache DomU2 machine, with a total number of 10000 requests. Each request is a file index.html = 5 MB. This file is located in /etc directory.

3.1.2 CPU performance between two virtual machines on different physical hosts

The same test is performed between DomU1 in the first machine and DomU2 of the second machine connected with gigabit interface. Those two computers are connected with twisted pair model line. The second physical machine is a clone of the first one. All the results are displayed in table 1.

Table 1 CPU processing between 2 VM in the same host and different hosts in LAN

Average time of CPU processing	Rate CPU processing	Between 2 VM in the same physical host	Between 2 VM in the different physical host
2,6 ms	80,1%	Yes	No
2,4 ms	79,9 %	No	Yes

As it looks there is a slight difference between the average time of CPU processing inside a physical machine and between different machines in a LAN.

3.1.3 CPU performance by increasing and decreasing of Memory in Virtual Machine

The third topic in CPU performance evaluation is the testing of CPU activities by increasing and decreasing the utilization of memory in virtual machine. Thus we will test the DomU1 Web Server. In apache 2.2.16 into the Web-server machine we have included test.c module. This module will serve as a tool in order to increase and decrease the virtual memory machine. At first we are located at /etc/apache2/apache.conf, then we have to install a tool: tool-sin apxs2 and compile it by command apxs2 –c –I –a mod_test.c. We can configure apache as multithreading process. Then we configure test file in /etc/ apache2/httpd.conf

```
<Location /test>
  SetHandler test-handler
</Location>
<Location /process_mem>
  SetHandler process_mem-handler
```

```
</Location>
<Location /increase_mem>
 SetHandler increase-handler
</Location>
<Location /decrease_mem>
 SetHandler decrease-handler
</Location>
```

The function test-handler will show something in web-pages. For every call in the increase-handler function, virtual memory of Apache grows up to 5 MB. The IP address of DomU1 is 192.168.1.1 and that of DomU2 is 192.168.1.2. In DomU2 is installed WAMP. From DomU1 we call 5 times the DomU2 by benchmark Httperf with command http://192.168.1.2/increase_mem and memory used by DomU2 will increase up to 25 MB. So this benchmark should manage memory utilization and CPU consumption in the Physical host. In the same way if we call http://192.168.1.2/decreas_mem , the virtual memory in DomU2 would decrease with 5 MB per time. We evaluate Response time and Page Fault Number by MemAccess Benchmark. We have presented all the results in table 2:

Table 2 Test results in Apache DomU2 based on two benchmarks Httperf and MemAccess with 5 MB for every iteration in mem_increase function for test.c module

Memory Utilization in Appache DomU2	Response time	Page Fault number	CPU Consuming
5 MB	0,036 ms	0	44 %
10 MB	0,040 ms	0	44 %
15 MB	0,047 ms	0	44 %
20 MB	0,059 ms	0	44 %
25 MB	0,941 ms	0	44 %

From the table 2 we take Response time, Page Fault numbers and CPU consuming if memory in apache web server increases from 5 MB to 25 M B. As it show the CPU consuming has a static value, evaluated to 44 %. Also there is no page faults and the response time has slightly growth. We got these results because memory utilization is too small comparison to each of Virtual Memory. Remember that DomU1 memory is 512 M B and DomU2 memory is 256 M B. If we repeat again the experiments by using httperf benchmark by modified test.c module from 5 MB to 100 MB and calling the increase_mem function for 5 times, we will get other results, which are presented in table 3.

Table 3 The results in Apache DomU2 based on two benchmarks Httperf and MemAccess with 100 MB for every iteration in mem_increase function for test.c module

Memory Utilization in Appache DomU2	Response time	Page Fault number	CPU Consuming
25 MB	0,941 ms	0	44 %
125 MB	7,967 ms	5	45 %
225 MB	15, 225 ms	9	45 %
325 MB	29,167 ms	21	47 %
425 MB	169,054 ms	22	47 %

From table 3 we see that response time is increases dramatically from 0,941 ms in 25 M B memory utilization to 169 ms in 425 MB memory utilization. This increase has occurs because memory utilization exceeds the Virtual memory needed from DomU2. As we wrote above, Memory of Virtual Machine in DomU2 is 256 MB. For the same reasons Page Fault increases dramatically when memory utilization exceeds 256 M B of virtual memory in DomU2. According to table 3 we see that Page Fault grew up from 9 Page fault to 21 page fault. When it reach to 256 MB memory size the page fault number has stability, because there is no matter how much memory is used above 256 MB limit size.
If we compare CPU utilization in table 3 and table 2, it is in the same level, approximately 45% (From 44 % to 47 %). This is because the exceeding of memory utilization does not affect significantly (or it has a slight effect) in CPU utilization. Now we have finished the experiments in evaluation of CPU performance.

3.2 Check the Network Performance

3.2.1 CPU Performance between two virtual machines in different physical hosts.

The second objective is to check the network performance in a cl uster system between three computers in the same LAN. There are two virtual machines in the client computer 1 and client computer 2.

The third computer will serves as a Storage Computer. Three computers are connected by UTP CAT 7 cable with Gigabit Ethernet. All computers have the same parameters and they can communicate with each other by UDP protocol with Gigabit Ethernet Switch. We used UDP protocol because:

- We are not interested in the reliability of sent information.
- Time sent and time response from one source machine to target one is more flexible and faster. These packets monitoring from Heartbeat benchmark and generated from Httperf benchmark.

Storage Computer will be associated with two problems:

I. There is a lower performance than SAN Storage (Storage Area Network)
II. There are no protection mechanisms, because there is no RAID Drive installed, which means there is no backup mechanism.
In figure 3 is shown the architecture of three computers connected by switch.

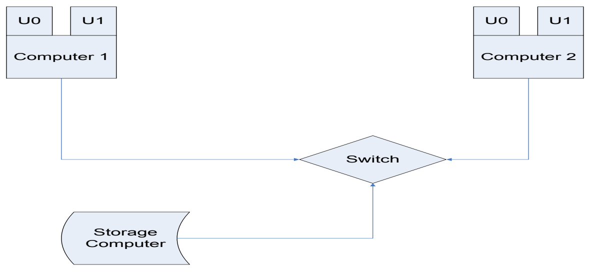

Fig. 3 The architecture of Cluster Systems with Physical Hosts and Virtual Machines above them.

The images of Virtual Machines should be put inside the Storage Computer. In the third computer we set up the iSCSI protocol. We use yast2 iscsi-server command. To test the installation, we have to execute the command cat /proc/net/iet/volume and we should get the volume name. During the installation of iSCSI protocol in remaining computers by command yast2 iscsi-client.
In the storage computer and hosts computers we have installed Ubuntu Server 10.04. In Guest Clients computers we have installed Ubuntu 10.10 Desktop and Win XP 32 bit. In DomU1 is installed Apache packet 2.0.3 and DomU2 is installed My-SQL-Server 5.0.9. Compile Xen and Dom0 for each computer.
At first we have to test the CPU performance between 2 Virtual Machines in different hosts computers.

We are using again the Httperf benchmark which now will generate 100, 200, 300, 400, 500 requests per second. CPU performance will show in Tab 4.

Table 4 CPU performance between two virtual machine in different hosts in the same LAN

Requests Number for second	CPU Performance
100 request/second	66,7 %
200 request/second	66,8 %
300 request/second	68,2 %
400 request/second	68,9 %
500 request/second	69,5 %

From table 4, note that CPU performance increases slightly when the number of requests generated from DomU1 in host 1 to DomU2 in host2 are increases from 100 requests to 500 requests per second.

3.2.2 Additive time caused from Warning Failure approach in a specific physical host.

The final experiment is the modification of init.d stop script in the storage computer. This script will upgrade the performance of live migration between DomU1 in host 1 and DomU2 in host 2. In the Storage computer we have installed My SQL Server 5.0.9. Also we should configure init.d stop script. This configuration will improve the live migration performance in two direction:
A. Virtual Machine will continue to migrate after a failure in the neighbour host into LAN
B. Xen Hypervisor installed on Storage Computer just detects the fail host, executes init.d stop script to all virtual machines which are located above the Physical host remaining. Thus the image of CPU Status and Memory Images will be transferred by Pre-copy or Post-copy iterative approaches to the healthy virtual machines. Immediately, physical machines will disconnect with failed physical machine, which means that they will not attempt to communicate with this machine in the future. We should emphasize that this situation can occur if the warning failure has introduced before. In other cases this mechanism doesn’t give any solution. In our example we give a warning failure for host 1 machine. It means that DomU1 in host 1 will transfer immediately to DomU1 of host 2. Thus the communications would being between DomU1 and

DomU2 above the host 2. This mechanism was made possible from init.d stop script in Storage Computer.
If init.d stop script is not modified it does not give any improvement performance, because if host 1 fails, the live migration technology would be not possible. The migration mechanism will transfer to Second Host and all services in Host 1 should restart again. Also the interruption time should take in consideration.To evaluate the time response in the network before and after live migration, from the shared storage computer, we send ICMP packets onto DomU1 Host1 and DomU2 Host2 by command:
ping –c 1000 -i 0.01 u1 ping –c 1000 -i 0.01 u2
To monitor the network performance we can use a benchmark called Heartbeat. We sent 1000 packets for an interval time equal to 10 seconds (100 packets for a second). For these 10 seconds we will simulate the warning failure of DomU1 and will repeat the experiment for DomU2 failure. For both cases the results are the same because the machines are clones. So the experiment will concentrate only on the first phase. If DomU1 fails the live migration total time from this machine to DomU2 will increases slowly. All this results are shown in table 5.

Table 5 Time migration with the warning failure machine approach.

Total Migration Time before failure by modifying init.d stop script in Xen	Total Migration Time after failure by modifying init.d stop script in Xen	Total Migration Time before failure without modifying init.d stop script in Xen	Total Migration Time after failure without modifying init.d stop script in Xen
10 seconds	10 seconds + 0,0456 sec	10 seconds	10 seconds + 2,062 sec

From Table 5 we present that the total time migration by modifying script is slower compares with no modification *init.d stop* script. In the first case, penalty time is approximately 40 ms but in the second case it is approximately 2 sec which means 50 times slower. In this way the performance in the second case is 50 times worst than the first case.

4. Conclusions and future Work

There are 5 conclusions from this paper:

I. The efficiency of CPU utilization does not change significantly if the packets are generated in the same physical host between two virtual machines or in different hosts in the same LAN. This is the merit of Xen para-virtualization [6], [7] technology Hypervisor.

II. The CPU consumption does not affect directly the memory growth. This is happen because Xen creates an isolation layer [3]. The memory ballooning approach is performed by including a test script in the apache module. In ballooning memory approach we use two functions: mem_increase and mem_decrease

III. Response time and Page Fault Number increases if memory utilization grows up.

IV. The CPU performance increases slightly if the request’s number from one VM to another one is increased.

V. The additive time caused from a warning failure in Live Migration approach is decreased significantly if we modify the init.d stop script in Xen Hypervisor.

To monitor the network performance we used the Heartbeat benchmark. To generate the request’s messages from one VM to another we used the Httperf benchmark. To monitor the memory utilization we used the MemAccess benchmark.

In the future work we will test the network performance, memory performance and CPU performance in WAN by using post-copy and pre-copy iteratively approach.

5. References

[1] Weiming Zhao, Zhenlin Wang, “Dynamic Memory Balancing for Virtual Machines”
[2] Hai Jin, Li Deng, Song Wu, Xuanhua Shi, Xiaodong Pan Hai Jin, Li Deng, Song Wu, Xuanhua Shi, Xiaodong Pan, “Live Virtual Machine “Migration with Adaptive Memory Compression”
[3] Kaushik Kumar Ram, Jose Renato Santos, Yoshio Turner, “Redisignin Xen`s Memory Sharing Mechanism for Safe and Efficient I/O Virtualization
[4] Michael Hines, Umesh Deshpande and Kartik Gopalan, “ Post-Copy Live Migration of Virtual Machines”
[5] Timothy Wood, Prashant Shenov, Arun Venkataramani, “Black-Box and Gray-Box strategies for virtual machine migration”
[6] en.wikipedia.org/wiki/Paravirtualization
[7] Andrew Tanenbaum, “Modern Operating System, Third Edition, Chap 8, p.574,”.

Igli TAFA. He is a pedagogue in Polytechnic University, in Computer Engineering Department. In 2008 he has finished the Master Thesis and now is PhD student. His PhD topic according to Virtual Machines direction.
Elinda KAJO (MECE). She is a pedagogue in Polytechnic University, in C omputer Engineering Department. She has finished the PhD thesis at 2004 in Object Oriented Programming direction.
Hakik PACI. He is a pedagogue in Polytechnic University, in Computer Engineering Department. He is a PhD student. His PhD topic according to Oracle DB System direction.