CHAPTER 1

INTRODUCTION

Virtualization addresses IT's most pressing challenge: the infrastructure sprawl that compels IT departments to channel 70 percent of their budget into maintenance, leaving scant resources for business-building innovation.

The difficulty stems from the architecture of today's X86 computers: they're designed to run just one operating system and application at a time. As a result, even small data centers have to deploy many servers, each operating at just five percent to 15 percent of capacity—highly inefficient by any standard.

Virtualization software solves the problem by enabling several operating systems and applications to run on one physical server or "host." Each self-contained "virtual machine" is isolated from the others, and uses as much of the host's computing resources as it requires.

1.1 Why do we use virtualization?

Virtualization is the single most effective way to reduce IT expenses while boosting efficiency and agility—not just for large enterprises, but for small and midsize businesses too. VMware virtualization lets you:

- Run multiple operating systems and applications on a single computer.
- Consolidate hardware to get vastly higher productivity from fewer servers.
- Save 50 percent or more on overall IT costs.
- Speed up and simplify IT management, maintenance, and the deployment of new applications.
- Migration from one physical machine to another without the loss of service.

CHAPTER 2
VIRTUAL MACHINE OVERVIEW

A virtual machine (VM) is a software implementation of a machine (i.e. a computer) that executes programs like a physical machine. Virtual machines are separated into two major classifications, based on their use and degree of correspondence to any real machine:

1. A system virtual machine provides a complete system platform which supports the execution of a complete operating system(OS). These usually emulate an existing architecture, and are built with the purpose of either providing a platform to run programs where the real hardware is not available for use (for example, executing on otherwise obsolete platforms), or of having multiple instances of virtual machines leading to more efficient use of computing resources, both in terms of energy consumption and cost effectiveness (known as hardware virtualization, the key to a cloud computing environment), or both.

2. A process virtual machine (also, language virtual machine) is designed to run a single program, which means that it supports a single process. Such virtual machines are usually closely suited to one or more programming languages and built with the purpose of providing program portability and flexibility (amongst other things). An essential characteristic of a virtual machine is that the software running inside is limited to the resources and abstractions provided by the virtual machine—it cannot break out of its virtual environment.

2.1 Virtual Machine Types

2.1.1 System Virtual Machine

The desire to run multiple operating systems was the original motivation for virtual machines, as it allowed time-sharing a single computer between several single-tasking Operation Systems. In some respects, a system virtual machine can be considered a generalization of the concept of virtual memory that historically preceded it. IBM's CP/CMS, the first systems to allow full virtualization, implemented time sharing by providing each user with a single-user operating system, the CMS. Unlike virtual memory, a system virtual machine allowed the user to use privileged instructions in their code. This approach had certain advantages, for instance it allowed users to add input/output devices not allowed by the standard system.

System virtual machine advantages:

- multiple OS environments can co-exist on the same computer, in strong isolation from each other
- the virtual machine can provide an instruction set architecture (ISA) that is somewhat different from that of the real machine
- Application provisioning, maintenance, high availability and disaster recovery.

2.1.2 Process Virtual Machines

A process VM, sometimes called an application virtual machine, or Managed Runtime Environment (MRE), runs as a normal application inside a host OS and supports a single process. It is created when that process is started and destroyed when it exits. Its purpose is to provide a platform-independent programming environment that abstracts away details of the underlying hardware or operating system, and allows a program to execute in the same way on any platform.

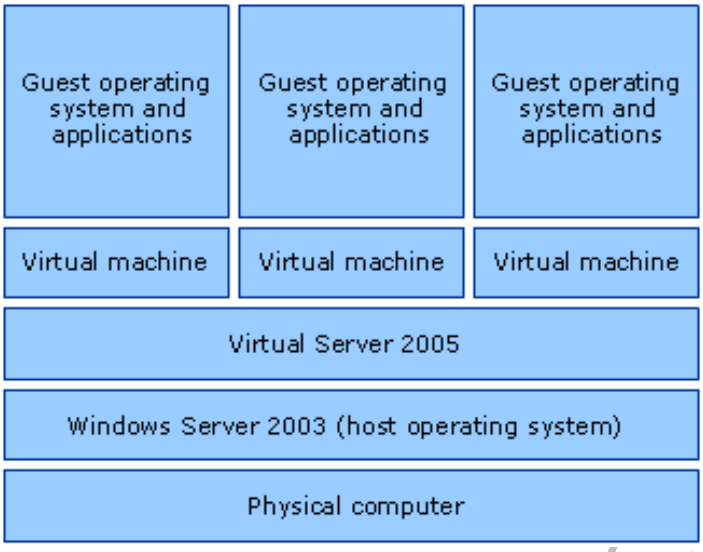
A process VM provides a high-level abstraction — that of a high-level programming language (compared to the low-level ISA abstraction of the system VM). Process VMs are implemented using an interpreter; performance comparable to compiled programming languages is achieved by the use of just-in-time compilation.

This type of VM has become popular with the Java programming language, which is implemented using the Java virtual machine. Other examples include the Parrot virtual machine, which serves as an abstraction layer for several interpreted languages, and the .NET Framework, which runs on a VM called the Common Language Runtime.

2.2 Virtual Machine Architecture

Most computers consist of multiple layers of hardware and software that operate together as a system. Hardware resources typically include a central processor, display, storage, networking, and other peripheral devices. Device drivers installed in the operating system control hardware resources, translating operating-system instructions into the specific device control language. Device drivers assume exclusive device ownership. For example, it is assumed that a video driver owns the video adapter exclusively. Any software application that calls the video adapter must interact with the hardware through the video driver.

Exclusive device ownership typically precludes running more than one operating system simultaneously on a computer. Virtual machine technology overcomes this limitation by redirecting interactions with device resources to lower system levels in such a way that higher-level application layers are unaffected. The Virtual Server 2005 virtual machine technology allows you to run multiple operating systems simultaneously on a single physical computer.



Virtual machine architecture

Starting from the bottom of the logical stack, the host operating system manages the physical computer. The Virtual Machine Monitor (VMM) virtualization layer manages virtual machines, providing the software infrastructure for hardware emulation. Each virtual machine consists of a set of virtualized devices.

All virtual machines run within a single process. Each virtual processor runs on its own thread. All software code running within the virtual machine runs in a separate VMM context. This context consists of an address space that is completely separate from any Windows process, or from any other virtual machine. In this sense, the architecture is stronger than a simple per-process design.

Finally, the guest operating system and applications run on the virtual machine as if they were running on physical hardware, rather than emulated hardware. When a guest operating system is running, the special-purpose VMM kernel manages the CPU and hardware during virtual machine operations, creating an isolated environment in which the guest operating system and applications run close to the hardware at the highest possible performance.

CHAPTER 3

VIRTUAL MACHINE MIGRATION OVERVIEW

Virtualization technology can play an important role to provide the multi-tenancy effectively. Virtualization gives an abstract view of hardware by means of virtual instance of multiple guest operating systems on a single host operating system. This virtual instance of operating system is known as virtual machine. The user can access the hardware in terms of virtual machines, which can simulate the hardware. As many virtual machines can be created on a single physical machine, depending on their capacity. It is clear that by proper management of virtual machines can improve the resource utilization and availability.

Virtualization allows a migration of virtual machine from one physical machine to another. Migration can be offline where user is halted till the virtual machine can resume on target machine. Another is live migration in which user is able to continuously execute during the migration. Migration of entire operating system and all of its applications as one unit (i.e. virtual machine) is less complex than process level migration. To avoid the dependency between the physical host machine and virtual machine, virtual machine monitor (hypervisor) is used as an interface. It is simply a software layer between host operating system and virtual operating system. One of the widely used virtual machine monitor is Xen [1]. It is a high performance resource managed virtual machine monitor, which allows various virtual machines to share common hardware in a safe and resource managed fashion, but without sacrificing the performance and functionality.

Migration of entire virtual machines means that in-memory state can be transferred in a consistent and efficient fashion. This includes internal states of kernel level and application level. Live virtual machine migration is an extremely powerful tool for the cloud managers.

3.1 DISADVANTAGES OF VIRTUAL MACHINE MIGRATION

The typical solution for migrating a running OS from one platform to another is to leverage virtualization technology. However, this technique requires an extra layer of abstraction that introduces several disadvantages.

- **Capability Lag.** VMMs typically expose "lowest common denominator" virtual devices to enhance portability. These devices typically fail to expose the highest performance features of the physical devices present. Further, to take full advantage of physical devices, up to three modules must work in concert: the device driver in the VMM, the virtual device model in the VMM, and the device driver in the guest OS. The difficulty inherent in ensuring this stack works together to provide efficient access to the device implies a high probability that a system performs less than optimally.
- **Software Management.** Although virtualization proponents claim that VMMs are sufficiently small to introduce little additional complexity into software management, virtualization typically does not reduce the total number of software components running on a system. Hence, there are more lines of code to manage, more patches to apply, etc.
- **Performance.** In many applications, virtualized performance is within an acceptable margin of native performance, and therefore, the additional layers of software introduced through virtualization are tolerated, but there are also cases where it is not. For example, Figure shows the performance overhead that virtualization imposes on a parallel robotics simulator. The application is based on MPI and makes frequent calls to barrier primitives. Running in a standard VM configuration, it suffers a 40% slowdown; even with a para-virtualized NIC, the slowdown is still 25%.

CHAPTER 4

RESOURCE OPTIMIZATION USING VIRTUAL MACHINE SWAPPING

4.1 INTRODUCTION

With the virtualization achieving high-level performance, the resource requirements of the applications in data centers is increasing at a tremendous cost. Resource management will play a key role in the next generation virtualization. Where traditional computing mainly focuses on resource cloud computing seeks to utilize resources with minimal wastage. We argue that this will require a new strategic plan towards dealing with cloud computing resources which are constrained.

Resource management requires strategies related to virtual machine placement and migration. These two techniques are capable of allocating and reallocating dynamic resources efficiently. A key to this work is the implementation of swapping technique for virtual machine migration for resource utilization. We consider memory as the resource requirement. There is two types of virtual machine movements transfer and swap Transfer moves a job from one machine to another, while swap exchanges two jobs assigned to different machines. We use the movements, transfer and move in our work. Most of today's virtual machine provisioning method swaps virtual machines without considering the resource optimization. This results in wasteful virtual machine swapping and resource wastage. While the focus of the paper is on resource optimization, we note that virtual machine swapping is considered as an important component in virtual machine placement and scheduling.

We focus this paper on the virtual machine swapping that supports virtual machine placement for efficient resource utilization. We describe our swapping model, and through a preliminary performance evaluation, demonstrate that resources can be effectively utilized with minimal wastage. The remainder of this abstract is organized as follows: In chapter 4.2 we describe the existing work in virtual machine scheduling and migration. Chapter 6.1 we present the algorithm for resource utilization with VM migration and swapping. In chapter 6.2 we

provide preliminary results of the effectiveness of our algorithm through simulation, and in chapter 6.3 we describe the future directions and preliminary conclusions that can be drawn from our work.

4.2 RELATED WORK

The virtual machine placement and migration on physical servers in cloud data centers has been well analyzed and studied. The impact of migration of virtual machines on power saving, server consolidation, load balancing has been studied on some of the research works. Some research works have been done recently in Server consolidation using virtual machine migration like performance overheads discussed in[6][7]. Huang et al. [8] has formulated a model that uses

Remote Direct Memory access to improve the efficiency of VM migration. The Sandpiper system model used by Wood et al.[9] make use of live migration technology to terminate hotspots in data centers. Other relevant work in the area of resource allocation, reservation and scheduling can be found in [10].

Most of the virtual machine migration techniques do not consider the swapping method for effective resource optimization. In this section we briefly discuss some of the virtual machine algorithms. Our swapping algorithm shares performance ideas and methodologies of AppLeS [I] and GrADS [2] .The high performance of process swapping has been discussed in NWS [3], Autopilot [4], MDS[5]. In [11] O.Sievert and H. Casanova et al. have presented a technique, called MPI Process Swapping. This swapping model chooses the best resource through process over allocation. The algorithms [12][13] helps in finding the best neighborhood for swapping. Our strategy is based on a combined model of virtual machine migration and swapping for highest resource utilization.

CHAPTER 5

VIRTUAL MACHINE MIGRATION and SWAPPING MODEL

This section presents the mathematical formulation of the algorithm. Objective of this formulation is to maximize the hardware resource utilization of servers in data centers. We start with a cloud computing environment consists of 's' physical servers on which 'V' virtual machines are hosted and running. Each virtual machine denotes one resource requirement (in our case memory is considered as the resource) from the user. . Assurne that the set of physical servers S=(sl, s2, s3 . . . . . sn) and the resource capacity of each server is considered as 100%.

Let V sm be the virtual machines in the S server (m=l ,2, ...) and j denotes the job.

The incoming request for a resource mj in a physical server should not exceed the maximum capacity of a server S as (eql)

$$m_j \geq [\ 1-\sum_{s=1}^{\infty}\sum_{m=1}^{\infty} Vsm] \qquad (1)$$

The incoming request mj should satisfy the condition (2) so that we can check the space availability for the new VM because physical machines should have enough space for the new request or else the scenario ends in instantiating a new physical machine.

$$Vsm_j\delta = \text{Max}\{\ Vs1_j, Vs2_j, Vs2_j, \ldots \ldots\} \qquad (2)$$

The Server which may result in maximum utilization after migrating a VM which is obtained after checking the various VM migration combinations is selected by using (eq 3)

$$Vsm_j+\sum Vm_j \leq 0.9,(m=1,2,\ldots),(s=1,2,\ldots.) \qquad (3)$$

$$\text{Let}\quad V_1=\text{Max}\ \{Vsm_j+\sum Vm_j\} \qquad (4)$$

The below expression (eq 5) assign the new virtual machine in the selected server which has the highest resource utilization

$$Vm_j = V_1 \quad (5)$$

If the above condition is false, then we go to the next level of swapping the virtual machine.

$$Vsm_j = Vsm_{j+1} \quad (6)$$

then

$$Vsm_i = Vsm_j \quad (7)$$

Each virtual machine in a server is swapped with another virtual machine in different server and the right combination of virtual machines is selected that results in highest server utilization

$$Vs_jm_j = 1 - \sum_{m=1}^{\infty} Vsm_i, (m=1,2,.....) \quad (8)$$

A temporary variable is assigned to place the swapped virtual machine temporarily as shown in the below (eq.9) and then the vm in the temporary server is placed back in the original server from where it was swapped as in (eq 10).

$$\text{Let } V = \text{Max}\{ Vsm_i, Vs_jm_j\} \quad (9)$$

$$Vs_jm_j = V \quad (10)$$

CHAPTER 6

PROPOSED ALGORITHM for VIRTUAL MACHINE SWAPPING

We propose here a virtual machine swapping model for resource allocation problem. The high level algorithm we propose performs an intelligent combination search for identifying the correct virtual machine for migration and swapping. A pseudo code of virtual machine swapping model is given in algorithm l. One of the distinguishing features of our algorithm when compared to other heuristics is the combined approach of virtual machine transfer (migration) and swapping. Additionally we used some parameters like network traffic, route, completion time of virtual machine for exploring the right neighborhood virtual machine for migration and swapping.

A virtual machine placement is based on the size of n jobs, virtual machine placement [i] indicates the machine where job j is placed to. We consider two types of virtual machine movement in our approach, transfer and swap Transfer the virtual machine from one server to another while swap interchanges two virtual machines between two different machines. It starts by finding a server that can afford the new job waiting in the queue and create a new virtual machine for the job and place in the selected server. If none of the servers can afford to place a new job, we conduct transfer and swap moves for placing the new job. It starts comparing the completion time of the virtual machines in a server and the migration time with the completion time of the new job. Then a virtual machine m is selected for a Server k and performs a suitable combination check with all the other virtual machines running in all servers.

The appropriate virtual machine combination is selected for migration by considering the parameters like network traffic, shortest route and completion time so that the server is fully utilized. If migration doesn't yield the expected result, then the algorithm considers virtual machine swapping as an alternative. The swap is considered as a possible solution only if the two physical servers have enough resources to place the other server's selected VM(s).

We have used a temporary server to place the VM before moving it to the target destination. The complete model of our approach is clearly shown in (fig 1).

6.1 ALGORITHM

```
New_vm_request()
{
        Server_list[]= get_the_server_who_can_afford_ it( vm size )
           if (server_list !=NULL)
          {
                  path _to_follow =
                 perform_validation( server_list[]);
                  Set_the_vm_to_dest();
         }
         Else
        {
                  setmigration = 1
                  vm_combtn[] = get_ al_the_vms_in_all_the_servers(all_servers[])
                  if( vm combtn[]! = NULL)
                 {
                          status_arr[]=get_ status of_servers when selected_vm_put()
                          target[]= get_the least(status arr[])
                          if (length( target[]» 1)
                                  path _to_follow = perform validation(server list[]);
                          Migrate( vm,source,destination)
                          Set_the_vm_to_dest();
                 }
                 Else
                 {
                          Set swapping= 1
                          Swap_vm[]=get_ combtn_of_all_ swaps()
                          Server list = Get_server_ list(swap vm[])
```

```
                if(server list> 1)
                        path _to _follow = perform validation(server list[]);
                Swap(vm,source,dest);
                Set_the_vm_to_dest();
                }
        }
}
Perform_validation( server list[])
{
        Identify all the paths leading to that server
        Identify the traffic in the network
        Return the least and shortest path
}
Migrate(vm,source,destinatn)
{
        if(required())
                 Ready the source and destination
                Ready the dest and migrate the VM
                Record dirty pages and Stop source svr
                Transfer dirty page and Start dest svr
        else
                Abort()
}
Swap (vm,source,destn)
{
        if (requird())
                Identify the temp server()
                Swap using temp server()
        else
                Abort()
}
```

```
requird()
{
        if (completion time > transfer time)
                Return 1
        else
                Return 0;
}
```

6.2 EXPERIMENTAL STUDY and RESULTS

Here we present the preliminary results of our virtual machine migration and swapping model, obtained through simulation. In the Figures 6.1-6.3 we demonstrate the impact of virtual machine swapping algorithm compared against virtual machine migration algorithm. We use trace data gathered from few hundreds of servers running different operating systems and different type of application running on them .A set of twenty simulations has been done ,using 2000 jobs selected from the trace data. Our data is assumed to be executed on five physical servers. Virtual machine is created randomly. The size of the jobs is determined by applying Poisson distribution. The execution time is uniformly distributed between 20 and 90 minutes. The experimental results prove that the algorithm results in maximum resource utilization of servers.

From Figures 6.1, 6.2, and 6.3 shows the result of simulation obtained for the algorithm. We can see that the introduction of virtual machine swapping results in an increase on resource utilization when compared to the virtual machine migration we use in the first part.

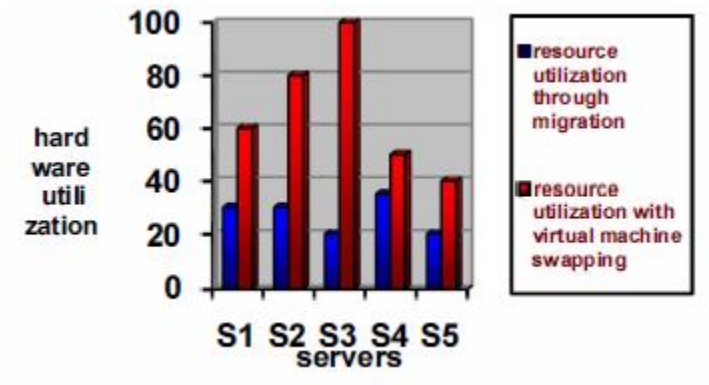


Fig 6.1 Resource utilization

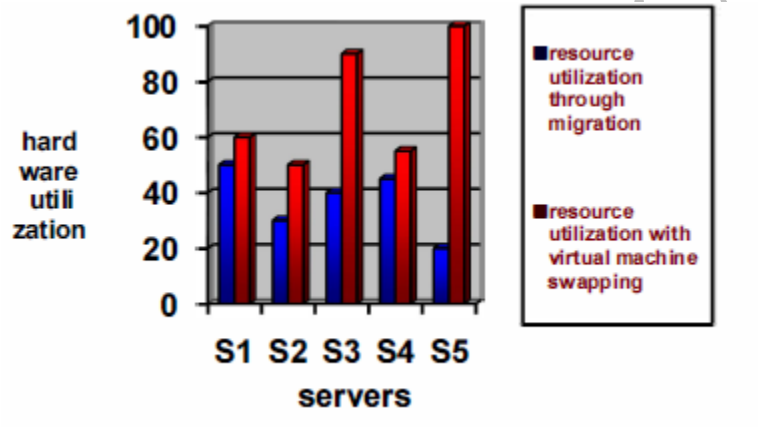


Fig 6.2 Resource utilization

Fig 6.3 Resource utilization

CONCLUSION

In this paper, we have evaluated the benefit of using virtual machine migration and swapping for improving the resource utilization in cloud data centers. The primary difference of our approach with the existing heuristics is that we have used the combined approach of migration and swapping of virtual machines for efficient resource allocation. We have proved that without virtual machine swapping, migration alone will not result in resource optimization. Our research offers scope for future research in VM swapping. In this paper, we have focused on VM swapping between only two servers by using a third temporary server. We can extend this work to VM swapping between multiple servers without a third server. We plan to incorporate our model in a real cloud data center with slight modifications.

REFERENCES

[1] F. Berman, R. Wolski, S. Figueira, 1. Schopf, and G.Shao.Application-Level Scheduling on Distributed heterogeneous networks. In Proceedings of Supercomputing 1996.

[2] K. Kennedy, M. Mazina, 1. Mellor-Crummey, K. Cooper, L.Torczon, F. Berman, A. Chien, H. Dail, O. Sievert, D. Angul0,I.Foster, D. Gannon, L. Johnsson, C. Kesselman, R. Aydt, D. Reed,1. Dongarra, S. Vadhiyar, and R. Wolski. Toward a Framework for Preparing and Executing Adaptive Grid Programs. In Proceedings of NSF Next Generation Systems Program Workshop(International Parallel and Distributed Pracessing Symposium 2002). Fort Lauderdale. FL, April 2002

.

[3] R. Wolski. Dynamically forecasting network performance using the Network Weather Service. Cluster Computing, 1(1):119-132, 1998.

[4] R. L. Ribler, J. S. Vetter, H. Simitci, and D. A. Reed. Autopilot Adaptive control of distributed applications. In HP DC, pages 172-179, 1998

[5] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W.Smith, and S. Tuecke. A Directory Service for Configuring Highperformance Distributed Computations. In Proceedings of the 6th IEEE Symp. on High Performance Distributed Computing, pages 365-375. IEEE Computer Society Press, 1997.

[6] P. Padala, X. Zhu, Z. Wang, etc., "Performance Evaluation of Virtualization Technologies for Server Consolidation",HPL 2007 .http://www.hpl.hp.com/techreports/2007/H PL-2007-59RI.html 1234

[7] A. Menon, J. R. Santos, Y. Turner, G. Janakiraman, and W. Zwaenepoel, "Diagnosing Performance Overheads in the Xen Virutal Machine Environment", VEE'05,2005. p.13-23

[8] w. Huang, Q. Gao, J. Liu, and D. Panda, "High performance virtual machine migration with RDMA over modem interconnects,"in Proceedings of the IEEE International Conference on Cluster Computing, 2008, pp. 11-20.

[9] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Blackbox and gray-box strategies for virtual machine migration,"in Prac. Networked Systems Design and Implementation,2007 ..

[10] 1.M. Schopf and L. Yang, "Using Predicted Variance for Conservative Scheduling on Shared Resources," in Grid Resource Management, eds. 1. Nabryski, JM. Schopf, and 1.Weglarz, Kluwer Academic, 2003/Virtualization.aspx (2009). [li] O. Sievert and H. Casanova. MPI process swapping: Architecture and experimental verification. Technical Report CS2003-0735, Dept. of Computer Science and Engineering, University of California, San Diego, 2003.

[12] R.K. Congram, C.N. Potts, S. L. van de Velde, An iterated dynasearch algorithm for the single machine total weighted tardiness scheduling problem, INFORMS Journal onComputing 14 (2002) 52-67.