Virtual Machines and the parallel processing

Johnny Chia

Computer Science Department  
San Jose State University  
San Jose, CA 95192  
415-541-8049

j.chial@hotmail.com

**ABSTRACT**

This paper is used to explain how Virtualization helps us to use multicores hardware more efficiently. It explain basics concepts about virtualization and it is focus on describing the hypervisor scheduler design goals, algorithms, co-scheduling load balancing and policy on Hyper-Threading. Additionally this paper will show some experiments done to evaluate various aspects of the CPU Scheduler.

This paper is based on VMware virtual machine software and on VMware vSphere ESXi 5.1 Hypervisor. The other technologies may vary.

For the implementation part, I have created a virtual environment test different configurations between VMware versions and compare this data to the data shown on my research.

# INTRODUCTION

Moore’s Law explains that explains that every two years the number of transistors in integrated circuits doubles. Clock rates limit has been reached. Instead of getting higher performance from higher clock rates, hardware has started gaining performance by increasing the numbers of processors.

The development of multicore hardware creates a new challenge, how to use the resources efficiently. Virtualization tries to solve this challenge by utilizing the full potential of the multicore processors and other hardware resources.

# CONVENTIONAL COMPUTING

Operating systems are usually assigned to one computer or server. Applications running on this OS have all the resources available in this computer or server. Here we face two problems. The first one is that there is not any software 100% parallel because of data dependency and the second one is that there are going to be times where the processors are going to be in idle.

Other problems non related to parallel processing are the inefficient uses of hardware, low scalability and high maintenance costs.

# VIRTUALIZATION

## Virtualization development

When one think about virtualization the first companies to come to our mid are VMware and Microsoft. However the first virtualization concept was developed by IBM in the 1960s to have better utilization of mainframes by logically partitioning them. These partitions allowed the mainframe computers to perform multiple tasks and applications at the same time.

During the 1980s and 1990s desktop computing and servers become available at a reasonable price. This made the virtualization technology being discarded. However, the IT infrastructure started becoming complex and included high maintenance costs, high infrastructure costs, and insufficient failure and disaster management.

VMWare created virtualization software for the x86 architecture. This idea came to address the new challenges during the 1990s. It worked so well that Microsoft acquired a software company Connectix to share the market with VMware; launching virtual PC in the 2004.

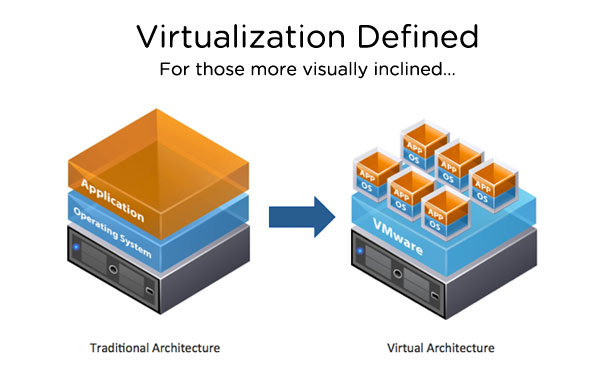
These two big companies are the gobal leaders in x86 virtualization.

## What is Hardware virtualization?

Hardware virtualization is the virtualization of computers or OS. It creates an abstract computing platform to hide the real physical characteristics of a platform from users. The software used to control this process is called Hypervisor.

In virtualization each Operating System is packed in a software container called Virtual Machine or VM. These virtual machines are completely isolated, but computing resources like CPU, memory RAM, storages and networking are pulled together and delivered dynamically with the help of the hypervisor (see Figure 1).

There are many benefits of virtualization. One of them is to run many applications on each server. Virtual machine encapsulates an entire machine, many applications and operating systems can be run on one host at the time. Other one is to increase server utilization allowing cost reduction [1].



**Figure 1 [1]. Graphic definition of virtualization**

Note that virtualization is not limited to a single computer, it can be run on clusters with external resources such as network attached storages.

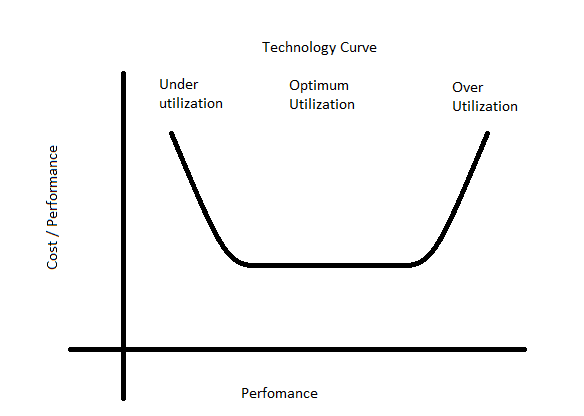
## VM allows going parallel

As explained before VM are completely isolated, this means that the data dependency between them is inexistent. Hence, this allows us to run each virtual machine in parallel.

For example, assuming we have a quad core machine running a Hypervisor with two virtual machines on it. The hypervisor assign two cores for each machine. Because there is no data dependency each VM can process their data in the assigned cores efficiently. In this easy example we just divided the resources between both VMs.

Also usually workloads peak hours are different for each application. Many servers stay in IDLE for long periods of time. For example, the accountant server has peak workloads during the mornings and the rest of the day is IDLE because the invoices arrived during the night and the sells server has peak workloads during the nights and IDLE the rest of the day. In this situation you only need the computing power of one server instead of two.

You might be wondering why to do this instead of buying two dual cores machines and run. The answer to this question is cost-efficient. We would like to stay in the optimum utilization section in the technology curve graph (see figure 2).



**Figure 2. Technology curve graph**

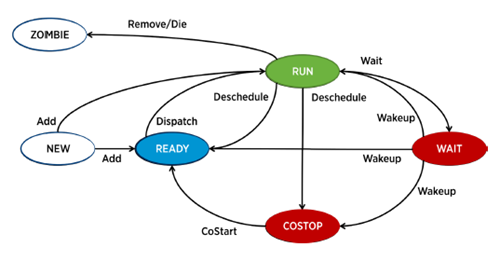
## VM hypervisor

“A hypervisor is a platform that allows multiple operating systems to run on a host computer at the same time [2]”. To do this with good performance the CPU scheduler is crucial.

### Hypervisor Scheduler

The scheduler has one main goal. “This role is to assign execution contexts to processors in a way that meets system objectives such as responsiveness, throughput, and utilization [3]”. There are four design goals: fairness, efficient throughput, responsiveness and scalability (for more information about goals see [4]).

Different to a traditional machine that consists of one or more CPUs, a VM consists of one or more vCPU (virtual CPU) world (a world is associated with a run state) on which guest instructions are executed. For instance, a 2 vCPU machine has 2 vCPU worlds. These are not the only worlds; there are other associated with the virtual machine to execute management tasks such as handling the keyboard and mouse, snapshots, and legacy I/O devices [4].



**Figure 3 [10]. Scheduler states**

In figure 3 we can appreciate how to follow the logical process of how a VM is scheduled on a CPU.

#### Proportional shared-based Algorithm

The scheduler has to choose which world is to be scheduled to a processor. If the processor is busy with another world, the scheduler hast to define if it needs to preempt the chosen world or not.

A word not necessarily consumes all the CPU due to CPU contention. This ratio of CPU consumption is used as the priority for the world. For example if a world has consumed less than its entitlement is considered high priority and most likely will be scheduled next. This priority is dynamically evaluated depending on scheduling, workload and system load. This is the key difference between shared-based and priority-based scheme [5].

The shared-based algorithm allows the user to control accurately the CPU allocation and can allocate different shares of CPU resources among groups of virtual machines. The CPU resource control is encapsulated and hierarchical. For example, consider a case where a user wants to divide computing power between different users and let the each user distribute the resources according to its own preference.

#### Relaxed Co-Scheduling

The term "Co-Scheduling" refers to a technique used in concurrent systems for scheduling related processes to run on different processors at the same time. It is applied to rim high-performance parallel applications [6].

An OS requires synchronous progress on all its CPUs. When running multiple VM the hypervisor must maintain this synchronization progress to prevent malfunction. In old versions the VM with multiple vCPUs were on schedule if there were enough physical CPUs available. For example, a VM with 8 vCPUs might not be scheduled even if there are seven idle physical CPUs.

To solve this issue Relaxed Co-Scheduling detects the difference in execution rates between two or more vCPUs associated with a single multiprocessor virtual machine. This difference is called skew. For instance suppose a multiprocessor virtual machine consists of multiple VCPUs, including VCPUs A, B, and C.  Suppose VCPU A is skewed, but VCPUs B and C are not skewed.  Since VCPU A is skewed, VCPU B can be scheduled to run only if VCPU A is also co-started.  This ensures that the skew between A and B will be reduced.  But note that VCPU C need not be co-started to run VCPU B.  As an optimization, the ESX scheduler will still try to co-start VCPU C opportunistically, but will not require this as a precondition for running VCPU B [6]. Relaxed co-scheduling significantly mitigated the CPU fragmentation problem.

#### Load Balancing

Load balancing is necessary to achieve high utilization and low scheduling latencies. However, migrating one process to another physical CPU has it cost. It is important to measure this cost in order to decide whether or not making the migration decision.

The scheduler makes the decision to migrate a process or not. One physical CPU can pull migrate a process or push migrate. To choose the best pair of pCPU in VMware there is a metric called Goodness metric. Goodness metric is calculated based on different criteria such as CPU load, Last-level cache, hyper-threading, the topological distance between physical CPUs, Co-scheduling and communication between scheduling.

Certainly using many factors in evaluating migration makes the scheduler take better decisions it also increases the cost of the algorithm. To solve this problem statistics are used to choose the best physical CPU, if it is a constant winner of the metric the current CPU is selected without being evaluated.

#### Hyper-Threading

Hyper-threading technology uses resources more efficiently, enabling multiple threads to run on each core. As performance feature it increases processor throughput, improving overall performance [7]. However, performance is limited because the computational resource is still one single physical CPU.

Due to this limit it will be unfair to a process to be always scheduled in a partial core and another to be always scheduled in a whole core. In this example consider that both have the same resource specification demand. To solve this problem the CPU scheduler charges CPU time partially if a process is scheduled on a partial core. If this situation persists a migration will occur to prevent it to be constantly behind.

## MEMORY SHARING

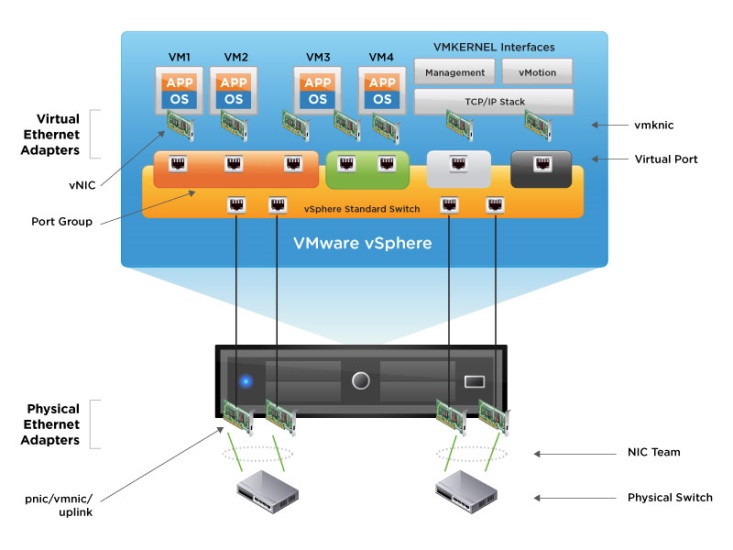
Random access memory or RAM is really important in the overall system performance. When a host is running out of ram it swaps part of itself to the hard disk paging files. This operation can significantly slow down the host.

Many workloads present the opportunity for sharing memory across virtual machines [9]. If two hosts are running the same operating system or similar applications it is inefficient to store the common data duplicated in the RAM.

The hypervisor constantly scans for memory sharing opportunity and uses a proprietary transparent technique to eliminate redundant memory copies, reducing the memory consumption for workloads. For example, imagine that we are running 2 virtual machines with windows on it which need 512MB RAM each. In independent computer you will need 1GB RAM so both can run with minimum requirements, but with memory sharing you may free up approximately 30% of memory, needing 700MB only.

## VIRTUAL MACHINES NETWORK

To communicate with each other Virtual machines uses virtual network running on the physical machine connected logically. There is a virtual switch who manages the traffic. These virtual switches works much like a physical Ethernet switch. They can connect to a physical network using physical adapters [10].



**Figure 4 [11]. Virtual network topology.**

In figure 4 we can appreciate in a visual way how the network topology works. Keep in mind that the speeds between physical and virtual networks will depend on the number and capacity of Physical Ethernet adapters.

Additionally if there is more than one virtual switch, it is possible not to connect one physical Ethernet adapter to two different virtual switches to enable load balancing features.

### Virtual Ethernet Adapter

Virtual machines can be configured with one or more virtual Ethernet adapter. The virtual host will see this adapter as common network interface cards.

### Virtual Switches

A virtual switch allows more than 1000 virtual machines to connect. This emulates a physical switch forwarding frames at the data link layer. Virtual switches have a good performance, and they do not add additional network load.

# INSTALL AND SETUP

We have already explained how Virtualization helps us to create high performance systems. Also how scheduler works in detail. Now it is time to explain how to install and setup a server. For this we will use VMware vSphere ESXi 5.1 client on a 12 Core Cisco Server with 48GB of RAM and 2.73TB Storage arrange on RAID 5.

The first thing we need to do is to download the software from <http://www.vmware.com/download/> and burn the ISO image. The installation is easy and straight forward. If you need more information you can check it here [8].

To manage the hypervisor we use a tool called VMware vSphere Client that can be downloaded and installed on any computer for remote management. This tool will let you create, edit and manage Virtual Machines very easily. Additionally, this provides real-time charts to measure performance.

This environment was tested in a real company, handling 4 application servers and one test server. The performance was outstanding and management costs were reduced significantly.

# CONCLUSION

Virtualization can be used as a great tool to create high-performance parallel servers. Most of the time the default configuration of the scheduler work fine, but the knowledge presented might when doing troubleshooting or system tuning.

Keep in mind that even if it allows us to run multiple Virtual Machines in single physical machines, performance will be limited by the physical computer. For example if you are running 10 virtual machines in a dual core server with 4GB of ram, do not expect high performance.

New ways such as cloud computing stated showing in scene to reduce IT and computing costs. However, Virtualization is still a good option to think about.

# REFERENCES

1. Virtualization Basics - How Virtualization Works. http://www.vmware.com/virtualization/virtualization-basics/how-virtualization-works.html
2. Virtual Machine Hardware, Options, and Resources Available to vSphere Virtual Machines<http://pubs.vmware.com/vsphere-4-esx-vcenter/index.jsp#configuring_virtual_machines/c_vm_hardware_options_resources_available_to_vm.html>
3. William Stallings, Operating Systems, Prentice Hall.
4. The CPU Scheduler in VMware vSphere 5.1 – Performance Study <http://www.vmware.com/files/pdf/techpaper/VMware-vSphere-CPU-Sched-Perf.pdf>
5. VMware vSphere 4: The CPU Scheduler in VMware ESX 4 <http://www.vmware.com/files/pdf/perf-vsphere-cpu_scheduler.pdf>
6. Co-scheduling SMP VMs in VMware ESX Server <https://communities.vmware.com/docs/DOC-4960>
7. Intel Hyper-Threading Technology <http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html>
8. vSphere Instalation and Setup – vSphere 5.1 <http://pubs.vmware.com/vsphere-51/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-51-installation-setup-guide.pdf>
9. VMware Workstation 5.5 - Memory Use on the Host <https://www.vmware.com/support/ws55/doc/ws_performance_mem_host.html>
10. vSphere Networking ESXi 5.0 vCenter Server 5.0 <http://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-50-networking-guide.pdf>
11. vSphere – About virtual networking <http://www.vmware.com/products/vsphere/features/network.html>
12. History of Virtualization, Oct 19, 2009 Edited Nov 13, 2013. <http://www.infobarrel.com/History_of_Virtualization>