Study of Virtualization & Management of Memory in Virtualized Systems

A Seminar Report
submitted in partial fulfillment of the
requirements for the degree of
Master of Technology
by

Aby Sam Ross

Computer Science & Engineering
Indian Institute of Technology Bombay
Mumbai 460076 (India)

April 2015

Abstract

Memory like other computing resources can be considered a scarce enough commodity. Multiplexing and management of this memory among various stakeholders in native computing environments has been well taken care of. But when it comes to virtualized environments it adds another layer of abstraction. This calls for new techniques, for partitioning and arbitrating memory among different VMs, that doesn't change the OS's perspective of the scheme of things and requires minimal changes. While the techniques for partitioning memory among different VMs in a virtualized setting has almost got standardized, there are still differing views and different approaches to managing memory. The new layer of abstraction introduced by virtualization adds considerable complexity in estimating the needs and utilization of memory. One cannot retrofit one strategy suitable for a particular setting to another and expect to get a clear picture of memory related parameters. This seminar is to understand different techniques of memory virtualization and to gain some insight into some of the existing management strategies.

Contents

Chapter 1

Introduction

1.1 Overview of Virtualization

Hardware resource abstraction and management have always been a major aim and challenge in electronic systems. With the advent of computers these tasks fell upon a huge piece of software called the Operating System. It became the responsibility of the OS to multiplex the resources among the different processes running in a system. An obvious illustration is that of CPU multiplexing, where the CPU is scheduled in time slices among various processes according to priority. The physical memory in a system as well is a resource. Memory also needs to be abstracted and suitable interfaces are to be provided for easy and efficient use. Traditionally this has been achieved through the concept of virtual memory coupled with paging or segmentation.

With advancements in technology the capabilities of physical resources increased and cost decreased. Processors and memory became faster, smaller chips with more capacity became available. But these advancements didn't percolate naturally into better utilization of these resources. Often these resources were underutilized. In order to get better utility from these resources various options of sharing these resources at a larger granularity was explored. Thus came into existence the concept of virtualization.

Virtualization increases the granularity of abstraction from individual resources to abstracting the entire set of hardware as a single unit or in other words virtualization abstracts the entire computer system. With this we could have more than one machine running on top of the existing computing hardware. These machines came to be called *Virtual Machines* or VM. In other words with the increase in granularity of abstraction

the unit of allocation to the abstraction increased from a process to an entire OS.
The ringmaster who runs the show in a Virtual Machine is still the humongous piece of code called the OS. But traditionally OSes were designed to have ownership and control of the underlying hardware. Virtualization now created a separation between the hardware and the OS.They were now relegated to the status of a *guest OS*. A single OS was no longer the sole owner and controller of the resources. The entire hardware had to be abstracted, interfaced and multiplexed among different OSes manning different VMs analogous to the way in which an OS enabled multiplexing of resources among different processes in a native system. This role was now taken up by a new, but no less hideous, software layer called the *Virtual Machine Monitor* (VMM) or the *Hypervisor*. That is now we have the hypervisor sitting on top of the hardware directly and above it we have different VMs.
The introduction of a new orchestrator, the Hypervisor, and relegation of a guest OS to a lesser privilege brought about new challenges. The guest OS in a VM had no longer complete control of the underlying resources to arbitrate efficiently among the processes running in it. But to the process local to a VM the guest OS was still the ringmaster who ran the show. Hence it became of paramount importance that the introduction of a software layer, the hypervisor, between the guest OS and the resources didn't break the equivalence view i.e. a process running in a VM should see no or little difference in running on a native vs. virtualized system. At the same time we also had to ensure that every VM stayed within its allocated bounds and guest OS operations had enough efficiency. These requirements of hypervisor design are formally stated in Popek and Goldberg (**?**).The hypervisor can choose from among the various strategies like instruction interpretation, trap & emulate, binary translation, para-virtualization, hardware assisted virtualization to provide the aforementioned requirements.
Once such a hypervisor is available efficient resource utilization and management comes next in order to meet the original design principles of virtualization. Memory like other resources will also be apportioned to the different VMs. But it is not necessary that all of the memory allocated to a VM will be in 100% use all the time. This gives us an opportunity to reallocate this unused memory to other VMs in need. But its sharing and management is not as simple as that of a flexible resource like CPU and nor are the effects of differences in the amounts of memory available that easily visible (**?**). The objective of this seminar is to get an understanding of the intricacies involved in this.

1.2 Scope of the Seminar

The objective of this seminar is not to delve deep into the implementation of virtualization as a whole but rather to concentrate on understanding how memory virtualization is achieved, the pros and cons of different techniques of memory virtualization, optimizations to it, challenges of memory management in virtualized systems, memory reallocation mechanisms, and some of the existing memory management strategies.

Chapter 2

Background

2.1 Memory Management in Native Systems

Memory management in non-virtualized native multiprogramming system is achieved using Virtual Memory. In multiprogrammed systems several programs are resident in memory at the same time. The memory management policy in such a system deals with protecting the memory of one program from another, loading a program into available space in main memory, de/allocating memory dynamically from/to programs.
While programs are complied and linked with addresses starting at 0 and CPU uses these addresses to access the binary, it isn't necessary (and is the not the case that) that a program will get physical memory with the same addresses or it is not even guaranteed that there will be enough space in the physical memory to load the entire binary of the program. Most of the times only the immediately needed part of the program is loaded onto the available physical memory. Hence a program is given the illusion that there is enough physical memory available to Hence the addresses generated by CPU needs to be translated into the corresponding physical addresses. The address generated by CPU is called *Virtual Address* or VA.