Article can be accessed online at http://www.publishingindia.com

On Demand Allocation of Memory and Virtual CPUs in KVM

Dinesh Madhavrao Bodewar*

Abstract

Virtualization is an emerging topic in today's world. Virtualization changes the way of thinking about IT infrastructure. Virtualization allows doing computing on virtual machine rather than physical one to save time and money. The most and relevant topic in virtualization is virtual operating system. There are different techniques to achieve virtualization of operating system. One of them is kernel based virtual machine i.e. KVM. KVM is a loadable Linux kernel module. It converts the Linux kernel into Hypervisor which is capable of creating virtual machine, and maintaining the virtual machine to run different types of virtual operating system. KVM uses all the features of Linux kernel to handle virtual machines. KVM uses hardware assisted virtualization from Intel and AMD. KVM uses a QUME module for emulation of Input/output devices. The virtual machine runs in user address space. KVM consider the virtual machine as a normal Linux process. Our proposed system tries to do two normal implementation changes in KVM memory management and allocation of virtual CPUs. The proposed system includes 1) allocate secondary memory to virtual machines at runtime as required. 2) Allocation of virtual CPUs at runtime on basic of result from algorithm. And add symmetric multiprocessing in virtual machines. This will speed up the working of guest operating system inside virtual environment.

Keywords: Operating System, CPU Scheduling, Process Control Block, Multiprocessing.

1. INTRODUCTION

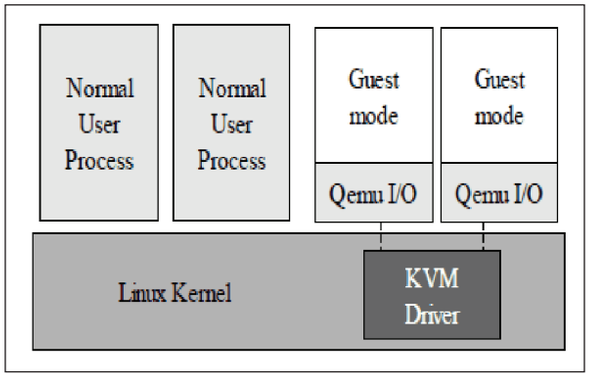
Kernel based virtual machine i.e. KVM is a loadable kernel module. KVM converts the Linux kernel into a hypervisor in other words KVM adds a virtual machine monitor capabilities to Linux kernel. There are two key design principles in KVM project 1) leverages all hardware assisted virtualization capabilities of Intel and AMD.[4] 2) uses a principle of "don't reinvent the wheel" concept[4] hypervisor is a specialized operating system. KVM project design is benefited from experience of Xen Hypervisor. In Xen Hypervisor implements all features for managing the virtual machine at its own. The disadvantage is that as technology changes and new features are added to kernel, Xen Hypervisor need to implement these features from scratch. This is done by adding custom patches. KVM is just as an accelerator it inherits the features of Linux Kernel to run virtual machines. The advantage of these features of KVM is that it uses the features of Linux Kernel without implementing them. For example, KVM uses the non uniform memory access support i.e. NUMA from Linux kernel but in Xen Hypervisor it need to implement NUMA support from scratch. Another key decision about KVM is that KVM incorporated into upstream Linux kernel. KVM code was submitted to Linux kernel community in December of 2006 and it come in existence from 2.6.20 kernel in January of 2007[4]. Hypervisor alone is not enough it needs other components like libvirt, QEMU, SELinux to make KVM successful. QEMU is only used in KVM. QEMU is used for input/output device emulation in virtual machines. QEMU is used by Hypervisor for creating virtual machines, stop, suspend, resume, and destroy virtual machines.

QEMU can handle virtual machines but this is not a good way, so for this libvirt is introduced. Libvirt is a c-library to interact with Hypervisor. It uses several management tool like virsh, virt-manager. virsh is a command line interface whereas virt-manager provides graphical user interface. Libvirt can create, start, stop, suspend, resume

* Department of Information Technology, D.Y. Patil College of Engineering Akurdi, Pune, Maharashtra, India.
E-mail: dbodewar1@gmail.com

and destroy the virtual machines. libvirt also take care of security in virtual machines, isolation between virtual machines.libvirt gives support for Linux control groups i.e. fine grained resource manager.

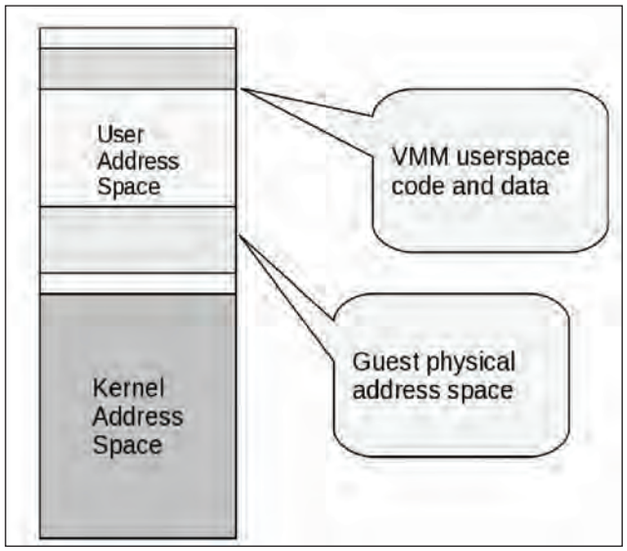
Figure 1. KVM Hypervisor



A. Memory Management:

KVM uses a powerful memory management feature from Linux kernel. A virtual machine is consider as a normal Linux process and virtual CPU is consider as thread only. Virtual machine executes at user address space. Virtual machine memory is stored as normal process memory. Linux have NUMA support for efficient access of large amount of memory.

Figure 2. Memory Management in KVM



B. Security

Security is major factor in hypervisor based virtualization. Our system security depends upon security provided to Hypervisor. Linux kernel provides Security Enhanced Linux i.e. SELinux project for mandatory access control and for policy enforcement. SELinux provides isolation of resources and processes running in Linux kernel. Svirt is a project built on SELinux to ensure that virtual machine resources cannot be accessed by another process or another virtual machine.

2. RELATED WORK

The core of the KVM is its memory management techniques and maintaining the workload among the virtual machines and increasing performance of the complete system.

In [3] it analyzes the performance of KVM, Xen and experiment present that KVM have a lower performance in CPU processing than the Xen Hypervisor. This performance difference is because of hardware emulation done by QEMU emulator. It clearly shown that Xen Hypervisor has a higher performance of reading and writing SAS discs than KVM.

In [3] it shows the way for evaluating the CPU consumption in OpenVZ. It creates a script in 'c' called *traceproc 1*. Hypervisor scans the Vstat file to trace the active and idle processes. traceproc1 does this by checking the wake up bit in process state register. If wake up bit is 1 the process is active and if it is 0 the process is in idle state.

In [1] a discussion on using multiple cores on a single processor. For fulfilling the increasing demand for high speed processor. The concept in [1] can be used in KVM based system to add symmetric multiprocessing and efficient use of physical CPU. In [1] virtual machine running under the Linux is considered as a normal operating system process. Virtual machines are scheduled as like scheduling a process.

In [4] have discussion on Xen Hypervisor and KVM. Xen hypervisor have mainly two components 1) Xen Hypervisor 2) Domain 0 and dom U virtual machine. Xen hypervisor performs core Hypervisor activities like virtualization of memory and CPUs, scheduling of virtual machines. Xen Hypervisor loads two virtual machines as Domain0 and domU. domU is unprivileged virtual machine.domain0 is privileged virtual machine having direct access to hardware and it provides device drivers. if there is any request for Input/output devices is directed towards domain0. Entire platform of Xen requires domain0 virtual machine to access hardware.

AMD developed the rapid virtualization indexing (RVI) also known as nested page tables. Intel provides extended page table feature within it. KVM uses Linux kernel feature called kernel same page merging i.e. KSM. KSM scans virtual machine memory and searches for identical pages if found then merges these pages into single page and share them between virtual machines. If any virtual machines try to change the shared pages it will be given its own private copy of that page.

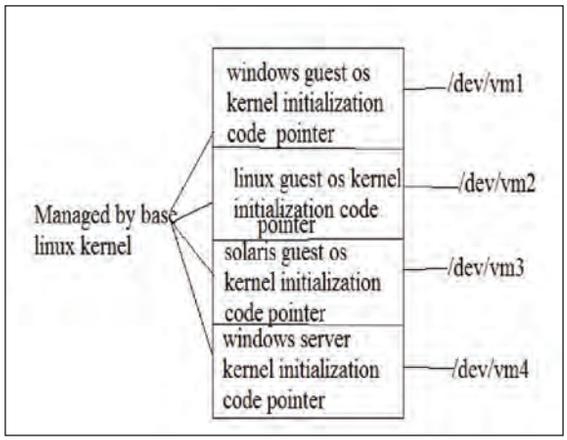
3. PROPOSED SYSTEM

In normal KVM based virtual machines and other hypervisor it need to preallocate a memory space and number of virtual CPUs at installation time. We are trying to eliminate the need of preallocation of memory and virtual CPUs. Here we are not using a Linux kernel as underlying base kernel upon which virtual machines are created. In the proposed system Linux kernel will provide a graphical or command line interface to interact with base kernel. While installing a new guest operating system it is installed at new file directory / dev/vm'n' with its unique identification number, and the corresponding entry is made in virtual operating system kernel table, which is maintained by base Linux kernel. The virtual operating system kernel table holds the pointer to initialization code of guest operating system. The guest operating system will look like an application on base Linux kernel.

Whenever a base kernel is turned on it loads the guest operating system kernel table in memory. When there is request for starting a particular guest operating system KVM matches the requested operating system id within guest operating system kernel stack and pointer to initialization code. Create a virtual machine environment allocates a memory space required for proper booting of requested guest operating system and handover the control to the kernel for proper boot up on itself.

In this case we are assigning a memory to guest operating system at run time as required. But this is not enough we need a new intermediate layer before assigning memory to virtual machine. The layer does some security checks. The intermediate layer is capable of making a privileged system call to KVM. The virtual machine cannot make the privileged call on its own. Any request from virtual machine goes to intermediate layer then to KVM and then only base Linux kernel will allocate the memory to the requested virtual machine. Base operating system has to manage the accounting of memory assigned to virtual machines.

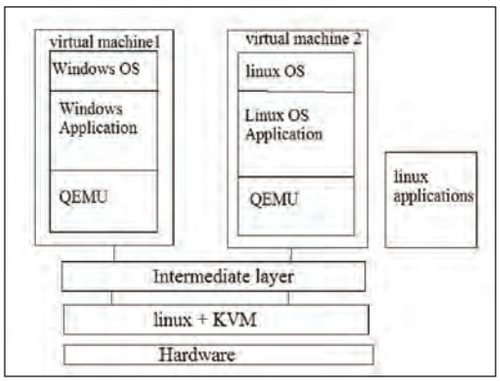
Figure 3. Guest OS Kernel Information Table



If a case that primary memory is not available or no free space due to working guest operating system. The intermediate layer checks for ideal process if any and apply some algorithms for virtual machines to free up space for requested operating system.

Virtual machines run in user address space. The memory management of the space provided to virtual machine is handled by the guest operating system running in the virtual machine itself. Guest operating system uses its own memory management features or it can use the allowed features of base

Figure 4. Proposed System Architectures



Linux kernel. The state information of each virtual machine is maintained by base Linux kernel, and does scheduling of virtual machines and maintaining accounting information of memory. Base Linux kernel will maintain accounting table for each virtual machines. When guest operating system turned off the memory values are stored in the table created for each virtual machine.

The intermediate layer is responsible for mapping of guest virtual memory to host physical memory. Because of this lot of burden from host operating system can be reduced. The live migration of virtual machines will become faster than earlier.

Do the same for allocating virtual CPUs. Virtual CPU is a normal thread for KVM. These threads of virtual CPU is created at run time depending upon the workload on each machine. Intermediate layer periodically apply an algorithms to check is there any increase in workload of virtual CPU, check a virtual CPU with minimum workload and send the result to KVM. Upon the result from intermediate layer KVM will decide to share the workload existing virtual CPUs of allocating a new virtual CPU.

So we can increase the performance of virtual machine by allocating virtual CPUs at run time.

4. CONCLUSION AND FUTURE SCOPE

In a proposed system we are keeping the same KVM based architecture with some changes like provision for allocating a memory and virtual CPUs to virtual machines at run time. Allocating a memory at run time and allowing the guest operating system to do memory management will help for faster live migration; speed up the working of guest operating system. Allocating the virtual CPUs at run time will improve the performance of overall system. It helps for workload sharing among virtual machines.

Different algorithms for workload sharing and deciding whether to assign a virtual CPUs to virtual machine or not. Several different techniques for isolation of virtual machines from each other

REFERENCES

[1] Tafa, I., Becirt, E., Pali, H., Kajo, E., & Xhuvani, A. (2011). *Evoluation of Transfer Time, CPU Consumption and Memory Utilization in Xen-PV, Xen-HVM, Open-VM, KVM-FV and KVM-PV*. 3rd International Conference on Intelligent Network and Collaboration System.

[2] Rosa, J. D. (2011). A Dell Technical White Paper on KVM Virtualization in RHEL 6 Made is Easy.

[3] Chem, G., & Gillen, A. (2011). White Paper on KVM for Server Virtualization: An Open Source Solution Comes of Age.

[4] Linux, R. H. (2011). Paper on Kernel based Virtual Machine.

[5] Petrovic, D., & Schiper, A. (2011). *Implementing Virtual Machine Replication: A Case Study using Xen and KVM*. 26th International Conference on Advanced Information of Networking and Application.

[6] Materneh, R. (2009). Multi microkernel operating system for multi-core processor. *Journal of Computer Science*, 5(7), 493-500.

[7] Ousterhout, J. K. (1982). *Scheduling Techniques for Concurrent System*. In Proceedings of the 3rd International Conference on Distributed Computing System (pp. 18-22).

[8] Sliberschatz, A. P., Galvin, B., & Gange, G. (2004). *Operating system concepts* (7thed). Hoboken, New Jersey: John Wiley and Sons.

[9] Tanenbaum, A. S. (1995). *Distributed operating system*. Prentice Hall.

[10] Muneer, H., & Rashid, K. (2006). SPE architecture for concurrent execution of OS Kernel and user code. *Information Technology Journal*, 5(1), 192-197.