CATEGORIZATION AND ELABORATION OF RECENT PROBLEMS IN VIRTUALIZATION

Kanika Pant
Roll No: 153050042
Dept. of Computer Science and Engineering
I.I.T. Bombay

April 22, 2016

Contents

1. Introduction

1.1 Evolution of Virtualization

Virtualization has its roots to the era of mainframe computers where time-sharing solutions were required to share the computer resources among large group of users for increasing the efficiency of both the users and expensive computer resources they share. Since then virtualization is evolving as a technology for wider range of applications.Data centers and cloud environment are extensively using virtualization technology in the present day scenario.

Therefore a lot of research work is going on to make virtualization technology perform better for the application domain in which it is being used. The recent work going on in the field of virtualization can be seen in various dimensions and aspects. These dimensions in virtualization can be explained as follows:

For virtualization to become a technology it is required to know its importance and benefits i.e. its applicability to provide a better solution than any existing solution.

If beneficial , how to implement this technology.For virtualization ,the first step in implementation is how to virtualize the hardware resources because as the definition of system virtualization says:Sharing resources such that there can be more than 1 machine for a single physical machine(hardware).There can be many ways of doing this resource virtualization.This virtualizing of resources has evolved in recent years towards and efficient way of doing it.The resources which require virtualization are:memory , I/O, CPU.

Once the resources are virtualized ,since resources are being shared , management of these resources is also required.Better management policies can be given for better performance.
When everything is setup and running then how these virtual machines are performing is an important question to be answered.

All these dimensions provide a lot information about virtualizaion , scope of improvement ,recent work and future scope in this field.

1.2 Scope of this seminar

In the course of this seminar I studied virtualization in detail.Recent research work going on in virtualization.The topic is explored in two different aspects:

- Categorizing the recent problems:

  Virtualization as a technology comes with several objectives,challenges,benefits and overheads.A lot of research is going on in this field so as to make this technology perform better by overcoming the challenges and minimizing the overheads.All this contribute to exploration of this of categorizing the recent problems in virtualization and understanding the problem description of problems in each category.

- In-Depth understanding of problems in memory management:

  This aspect explores understanding the problem description as well as solution of the recent problems in memory management category.

2. Categorization of problems

The area of virtualization is wide and open for developments and improvements.Therefore there can be different ways of categorization. This chapter covers the categorization of the recent problems.The reason for the existence of the categories. The various categories in which the recent problems are divided can be given as:

1. Resource Virtualization
2. Resource Management
3. Applications and Use-cases due to virtualization
4. Performance implications of virtualization

The next few sections will give the insight for the various categories presented.

2.1 Resource virtualization

System virtualization requires different hardware resources to be virtualized.These resources include:

- Memory
- CPU
- I/O Devices
- Network Devices

2.1.1 Memory Virtualization

Reasons for this category:

- Extending virtualization to cross ISA can prove useful in scenarios like recent introduction of virtual smart phones for iOS where mobile applications run on VMs or mobile application development on x86 platform.
- There can be vast range of applications that can be running on virtual machines.Performance for such applications is dependent on how memory is virtualized.

There are challenges in already present solutions for memory virtualization to perform better in these situations.

2.1.2 I/O Virtualization combined with CPU virtualization

The motivation behind this category can be explained in terms of problems which are faced due to I/O virtualization. Following are the problems :

- I/O device interrupt handling in virtual environment.
- I/O virtualization in para-virtual environment.
- Data piping(data transfer from one I/O to another I/O device) where data does not require any changes from the application with virtualized devices.
- I/O device interrupt coalescing in virtualization.

2.2 Resource Management

2.2.1 Memory Management

Memory overcommittment requires the need of memory management.Memory management policies include low level techniques of memory reclamation. The existing reclamation techniques can have problems and require improvements or a particular reclamation can be better over other.

2.2.2 I/O Management

Software based network virtualization and hardware based network virtualization have pros and cons.A higher level policy is required for addressing these issues. Performance

2.2.3 Data-center Level Management

Server consolidation is widely used in data centers for efficiently utilizing resources of the highly capable physical servers. Fault-tolerance and load balancing are also important issues to be considered in data-centers. There are techniques such as live migration and replication which play key role in handling above issues.Check pointing is the underlying technique for live migration and replication.Therefore it is required to consider certain things:

- How fast check-pointing can be done.
- How fast checkpointed memory can be restored.
- Storing virtual machine snapshots in a distributed file system with the aim to achieve high availability with low storage usage.

More issues associated with live migration

- Fast and bandwidth efficient live migration
- Extending live migration over WAN.
- Migrating related VMs in multi-tier application.

2.3 Application and Use-cases of virtualization

Virtualization can be useful in remote desktopping.Earlier remote desktopping tools work by streaming the display from the remote server to the users client computer.The network bandwidth requirement of this display stream is very high.Different optimizations like compression can be done for reducing the bandwidth requirement but still all these optimization algorithms require significant CPU cycles. Virtualization can help overcome these challenges.

Virtualization can be helpful for providing security and reliability to VMs using the fact that it provides isolation and interposition i.e. control.

2.4 Performance Implications of virtualization

The main contribution of virtualization is efficient hardware utilization. But virtualization comes with complexity of implementation and overhead. Using nested file systems in virtualized environment would impact performance.Analyzing the performance impact would help make better decisions.

3. Background

This chapter is for understanding the already present solutions/techniques in each category.The way things are handled.

3.1 Memory Virtualization

Physical memory is an important resource of the system therefore it needs to be used efficiently.In virtualized scenario each virtual machine is given a perception of physical memory with pseudo contiguous physical memory.The guest OS in the VM is allowed to maintain its own page tables for this pseudo physical memory.This adds an extra layer in address translation i.e. (guest virtual address)− >(guest physical address) and then (guest physical address)− >(machine physical address).MMU is responsible for this address translation There are various ways of virtualizing MMU so that overhead can be minimized.Existing methodologies of virtualizing MMU are:

- Shadow paging

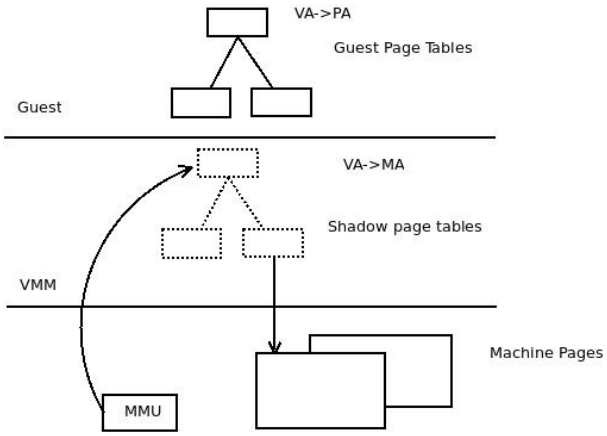

Figure 3.1: Shadow Page Tables

  Shadow paging mechanism is a software MMU virtualization mechanism.for each process in a VM, just like a guest OS maintains a page table, the hypervisor also maintains a page table called the Shadow Page Table.The shadow page table contains the direct mapping from guest physical to machine physical.Whenever there is a access to CR3 register by guest operating system that access is trapped by the hypervisor and hypervisor loads that CR3 with the base address of shadow page table.

- Direct Mapping

  Direct mapping is used with paravirtualized systems.Guest OS is modified as a part of MMU virtualization.When the guest page tables are updated then a hypercall is initiated from guest OS to hypervisor.The guest page tables are updated with direct mapping from guest virtual address to machine physical address which are hypervisor validated.

- Nested Page tables or Extended Page tables

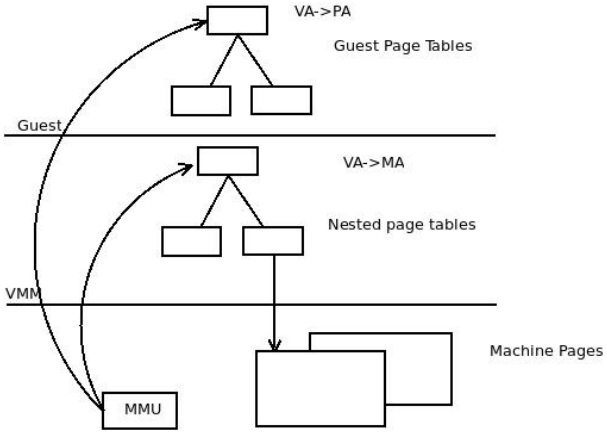

Figure 3.2: Nested Page Tables

Extensions to the hardware in the form extended page tables or nested page tables (EPT or NPT) enabled a new mechanism to achieve memory virtualization called hardware assisted MMU virtualization.There is no hypervisor intervention in this case.Both hypervisor and guest have their own set of CR3 registers.The guest OS maintain its own set of page tables.The nested page tables are maintained by hypervisor.Guest OS memory reference results in 2Dpage table walk-first in the guest page table to get the guest physical address and next in the nested page table to get the machine physical address.

3.2 I/O Virtualization

I/O virtualization is different from CPU virtualization and memory virtualization.I/O devices have different behaviour. They are slow as compared to memory and CPU.Non-determinism is also associated with them.I/O events take place asynchronously.OS also uses abstraction of device drivers to communicate with them.There is no such thing as device partitioning with I/O devices because only save and restore state wont result in device partitioning unless asynchronous nature of the I/O events is handled.Therefore I/O virtualization is handled differently.There are various models of I/O virtualization:

- Device Emulation model

  In this model a virtual device is exposed to virtual machines.the existing device drivers in the guest OS can be used.But there is an overhead of translating instructions for respective I/O device.

- Para-Virtualized Split Device Model
  In this model device drivers are split into 2 parts:
  - Front-end driver:
    This Front-end driver is present in the guest OS.
  - Back-end driver:
    This back-end driver is present in a privileged VM called as Driver domain. This model requires changes to the guest OS.There is no involvement of VMM in the device access

- I/O device sharing model:
  In this model I/O device itself supports multiple functional interfaces. These interfaces can be assigned to various VMs.SR-IOV comes under this model.

- Direct Device Assignment Model:

  Device driver in the guest Os can directly access the hardware i.e. physical device. There is minimum involvement of VMM but this requires additional hardware support. But there is no sharing of physical device.

3.3 I/O Device Interrupts in Virtual Machine

Flow of how I/O request is handled in virtualized system.This flow will cover both way process from application to device and vice-versa.

1. Application running within virtual machine issues an I/O request.
2. The I/O stacks in guest OS processes this request.
3. A device driver in the guest OS issues this request to a virtual device. This is a privileged instruction and hence, gets trapped by the hypervisor.
4. Hypervisor schedules the requests from multiple virtual machines onto an underlying shared physical I/O device, usually via another device driver maintained by the hypervisor.
5. When physical device finishes processing the I/O request, the two stacks must be traversed again but in reverse order.
6. The actual device posts a physical completion interrupt which is handled by the hypervisor.
7. Hypervisor determines which virtual machine is associated with the completion and notifies it by posting a virtual interrupt for the virtual device managed by the guest OS.

4. Description of problems in each category

4.1 Resource Virtualization

4.1.1 Memory Virtualization

I came across the following problems in this category:

- Nested Page tables described in the previous chapter can be quite useful with applications having good memory access locality because there will be less TLB misses but if the applications are with poor access locality then a TLB miss would convert into 2 dimensional page table walk thus increasing the latency and degrading the performance.The paper [7] tries to solve this problem.
- Tesseract: Reconciling Guest I/O and Hypervisor Swapping in a VM
  With server consolidation memory is over committed and therefore it is required by the hypervisor to dynamically reclaim memory.There are 3 types of memory reclamation techniques:

  1. Ballooning
  2. Content based page sharing
  3. Host level swapping

  Host level swapping is used in rear scenarios and it can cause serious performance penalties.If host is under severe memory pressure then it tries to reclaim memory through host-level swapping.It reclaims certain pages by swapping those pages to the disk. Now after sometime guest is also under memory pressure and tries to swap the same pages which were swapped by host.In this case hypervisor has to swap-in those pages so that guest can swap them out again.This increases the I/O latency and degrades performance.The main challenge here is to remove duplicate work of swapping the same page. The paper [3] tries to solve this problem.

4.1.2 I/O Virtualization And CPU Virtualization

The problems in this category are:

- There are various I/O virtualization models which work differently and have both advantages and disadvantages.One of the I/O virtualization technique which can be used is direct device assignment.Direct device assignment proves quite advantageous in terms of I/O performance because it does away with the intermediary overhead caused by hypervisor intervention which is there with other I/O virtualization techniques.But this direct device assignment negate some of the benefits of virtualization and also suffers from some deficiencies which limits its applicability.The possible deficiencies could be:
  - Restricting memory overcommittment by reducing the amount of memory that can be reclaimed dynamically.This is calles page pinning where guest pages are pinned which means they cannot be reclaimed.
  - IOMMU(I/O memory management unit), which plays a significant role while device is performing DMA to restrict DMA operations on specific memory locations.In this virtualized scenario hypervisor intervention is there for this IOMMU functionality for guest OS(paravirtualization). But direct device assignment is removing hypervisor from the scene and therefore is no control over the validation of IOVA(I/O virtual address).This can cost for security of certain specific

regions of memory which a device can directly access and change it.This poses a possible threat to device driver code.
Overcoming these deficiencies and getting full benefits from direct device assignment.The paper [2] is trying to solve this problem.

- Data piping is scenario when an application is trying to serve some client request by accessing storage disk and then sending those contents to network device without making any change to the contents from the disk.This requires to first go through the entire I/O stack for transfer of data from disk to application and then crossing the network stack for transfer the same data from application to network device.This increases I/O latency due to so much processing.
  When such applications run on virtual machines this I/O latency further increases as devices are virtualized and there is an extra layer of abstraction which adds up to the I/O processing delay.One more reason for this increase in I/O processing delay is because of CPU virtualization.CPU virtualization makes the VMs to run turn-wise thus VM scheduling is required.Therefore there is no guarantee that the interrupt will be immediately processed because may be at the time interrupt for a VM that VM may not be running. therefore this would increase the delay and degrade the performance. So the challenge in such situation is to offload the work to VMM layer by having the knowledge whether a particular data requires changes from the application.The paper [6] tries to solve this problem.

- Server consolidation is the primary benefit of virtualization.Different hardware resources are virtualized.Among the various resources memory and CPU are the resources which dictates the level of server consolidation.Memory is strictly partitioned among VMs but there are dynamic memory reclamation techniques for adjusting the amount of memory available for each VM.Therefore overcommittment is possible with memory.CPU is shared among VMs.Each VM is assigned one or more than one virtual CPUs.CPU scheduling is required for VMs which is done by hypervisor.An over committed situation is created as the number of vCPUs are larger than the number of pCPUs.Therefore even if the vCPU is ready for execution there is no pCPU available as the VM is not scheduled and it has to wait. This situation has a direct impact on device interrupt processing.I/O processing in modern OSes takes place in two stages:

  - In the first stage synchronous processing in an IRQ(interrupt request) context takes place in the kernel and the data(network data or disk reads) is buffered in kernel buffers.
  - In the second stage the data is copied from kernel buffer to user-level buffer by the application in an asynchronous manner.

  The IRQ processing is done immediately in dedicated system where the Os is running directly on physical machine or there is a dedicated core for a given VM.But if CPU is shared among the VMs and there is no dedicated core for a VM,the IRQ processing may be significantly delayed because The VM may not be scheduled at the time of I/O event.

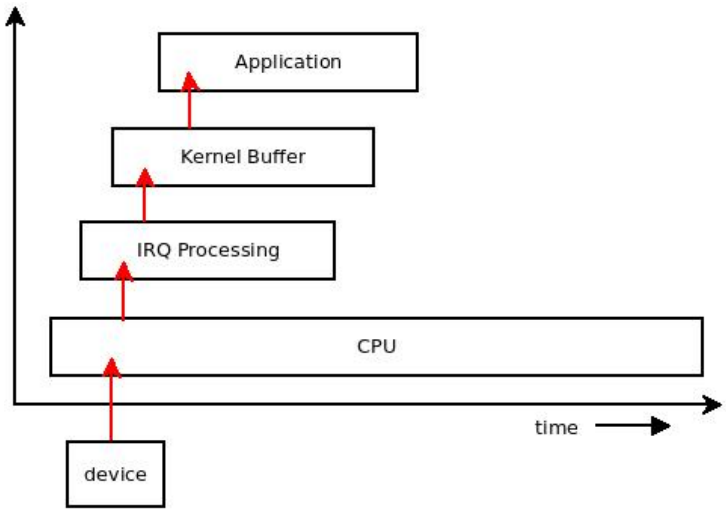

Figure 4.1: IRQ Processing with Dedicated Physical Resources

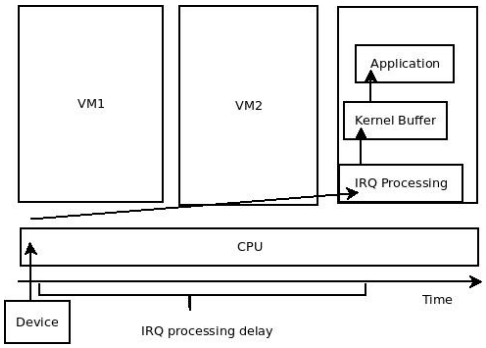

Figure 4.2: IRQ processing with VMs sharing CPU

Consider some scenarios where we see the direct impact of this problem.Suppose some TCP packets are arriving for some VM and they are getting buffered in shared buffer between hypervisor and guest VM.But that VM is not scheduled on any physical CPU at that time. This decrease the TCP throughput because the packets will be processed when that VM gets scheduled and this would delay ACK generation for those TCP packets.With UDP packets there is no time sensitive ACK mechanism therefore packets will be continuously generated for that UDP and connection.This overflows the shared buffer and packets start getting dropped thus reducing the UDP throughput.With disk I/O suppose application continuously writes to the kernel memory and after sometime the kernel memory gets full and write to disk is required.Then IRQ is contacted for flushing the data to the disk in the background.But if the VM is not scheduled then this increases the I/O processing delay.Diagrams for the same are given below:

The paper [13] tries to solve this problem.

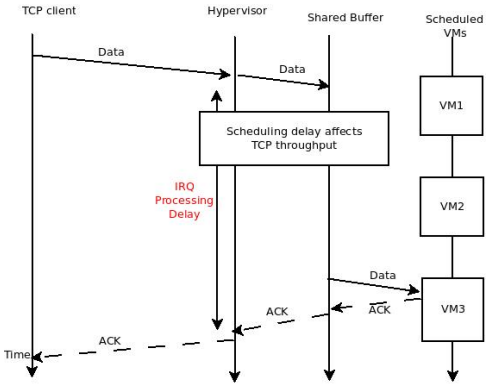

Figure 4.3: TCP packets

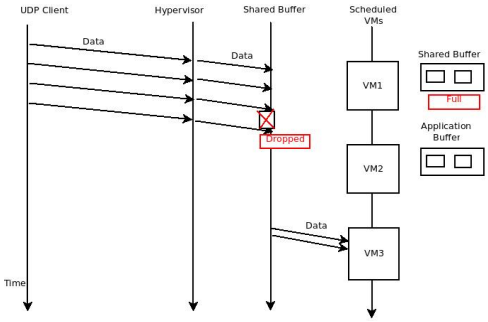

Figure 4.4: UDP Packets

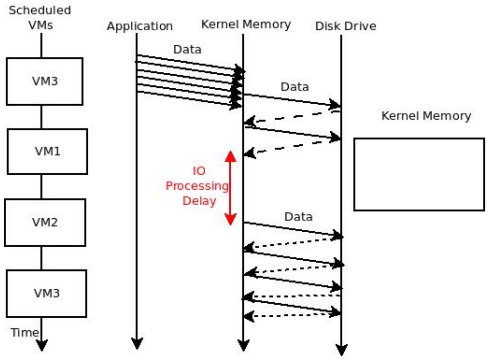

Figure 4.5: Disk Write

4.2 Resource Management

4.2.1 Datacenter level Management

Problems in this category are:

- VM migration is a benefit of virtualization which proves quite useful in load balancing and maintenance situations.VM live migration takes place between two closely related physical hosts within a cluster where hosts share the same storage device and common ethernet subnet facilitating network mobility.With the improvement in speeds of ethernet upto 100Gbps and high performance SSDs have scaled the VM deployment to 10,000 over multiple sites and removed local sharing of storage.With all these facilities migration is not just a local area affair ,it has moved to cross the local boundaries to wide area.The solution for this wide area migration using the existing live migration is a naive solution because live migration is done at the assurance that the downtime of service is minimized.Now if we use the ad-hoc techniques downtime increases as metro area links are in the range of few gigabits and as the distance increases latency also increases and throughput decreases.Network bandwidth is also under-utilized. One more issue with these ad-hoc migration

techniques is migrating storage.The approach is to first doing storage migration to a scratch space which is a shared storage using network file system from the source host and then migrating it from scratch to destination.This certainly increases the downtime and also there is no coordination between this storage migration and memory-device state movement. With all this running the VM over WAN while its disk located on the scratch space degrades performance.Since this is wide area migration migration would take place hop by hop and during all this network failure happens then since there is no coordination.Disk would be on one side and memory on the other.
Thus current solutions have a lot of challenges.The paper [9] is trying to solve this problem.

- Web applications can follow multi-tiered architecture as it can result into highly scalable and reliable web applications.In virtualized environment the architecture is deployed such that it requires interaction of multiple VMs.This is because we can have web server ,application server and database server all running on different VMs. For large corporations multi-tier applications could be deployed in multiple data centers. For maintenance or load balance activity VMs are required to be migrated.Since these multi-tier applications are highly interactive therefore if migration is done intelligently i.e. suppose after migration these multi-tier components get split across high latency path or a congested network then it would severely affect application's performance. This is not handled with conventional migration technique.The paper [1] is trying to solve this problem.

- Virtualization comes with the benefit of live migration or replication.All this is possible because of technique in which the volatile state is recorded and later when required it is restored.This technique is called check pointing.The saved state is called checkpoint or snapshots. For getting the snapshot of a virtual machine it is required to store the CPU,device and memory state.Memory snapshot is dominant part of the snapshot because the contents of the memory are stored to disk.With increasing size of the memory in VM the size of the snapshot does play a important part as it is required minimize the size of the snapshot because if for example snapshotting is used for live migration the a lot of bandwidth is required.One more issue is that this snapshot is stored in the disk and a lot of space will be used for this.Time is also a crucial aspect and will increase with checkpoint size. The basic catch in this situation is that it is not required store all the contents of the VM memory if same content data is already present in the external storage. The main challenge is to exclude the pages which are already present on the external storage. This way checkpoint time and space for checkpoint reduces.The paper [10] tries to solve this problem.

- Checkpointing described in the previous problem is a save and restore mechanism.This save and restore has several benefits such as suspending a VM if not used to save power then restarting it again when required.This can also be used for backup and fault tolerance. Challenges for saving the state are described in the previous problem.There are challenges for restore also as restoration takes place from persistent storage which is a very slow device therefore it cannot be restored quickly.One simple technique can be to restore all the memory contents at once in the beginning. The restoration time will linearly increase with the size of VM.This technique is called eager restore.There could be one more variation to this which is called lazy restore.It says load only device and CPU state initially and restore memory contents in the background while the VM runs.This certainly start VM faster.But for the contents which are not restored ,the page for the contents must be faulted in. This would pause the VM execution until the page has been retrieved.This have a significant performance impact as response time will increase for user.
  This shows that a good restoration technique is required for better performance.The paper [15] tries to solve this problem.

4.2.2 I/O Management

The problems in this category are:

- Disk is slow device and requires good policies for handling disk requests.Disk schedulers are required for this purpose.Disk schedulers works with the objective of reducing disk head movement i.e. disk seeks because disk seeks become bottleneck for disk performance.This is done by exploiting spatial locality in the requests.The conventional disk schedulers which are used are basically work conserving.In this work conserving approach the requests which are pending are processed first even if the pending requests are far away from current disk head.This would certainly degrade performance. The paper [14] tries to solve the problem.

- Resources are shared among various virtual machines when running on the same physical server. Consider I/O stack where there is a possibility of maximum interference when contended by various VMs, all application workloads collocated on a host issue their requests to a shared file-system, then there is a global page cache(DRAM) which holds recently used data blocks but there is no distinction between workloads.This page cache is used by file-system for all the requests.After this moving down the stack all these requests are served by giving access to the disk backing the file-system through a shared and contended path for access. This shows performance interference with shared resources which ultimately degrades performance. The paper [12] tries to solve this problem.

- Virtualization involves virtualizing hardware resources.Of all the resources including CPU ,memory and I/O , I/O virtualization is different because I/O devices are quite slow and I/O events are asynchronous in nature therefore some mechanism is required for delivering these events to virtual machines.Another issue with I/O devices they cannot be partitioned along time and space axes.There are various techniques for I/O virtualization which are already discussed in previous chapter. There were software approaches as well as hardware assisted approaches for I/O virtualization for network devices. A brief introduction to hardware approach.
  With the advancements in NIC design facilitates with in-device partitioning capabilities.Single root IO Virtualization is approach with such facilities in which NIC device provides with multiple virtual interfaces called as Virtual functions(VF) for the virtual machine.A special driver is required inside the guest OS which can interface with these VF.This way virtual machines directly communicate with the hardware device giving high I/O performance and minimizing overheads.But This takes of the control from hypervisors and complicate virtualization enabled techniques like migration and cloning.It also restricts the number of VMs which can run on the physical server with the available number of virtual functions in the hardware.
  With software approach of device emulation ,it comes with a lot of overhead as all the device related operations are trapped and controlled by the hypervisor.
  Another technique which is split device model ,although performs better than device emulation but still incurs overhead of copying packets between guests and the privileged domain.
  The paper [5] tries to solve this problem.

4.2.3 Memory Management

The problems are explained in detail in detail in later chapters. The problems in this category are:

- Applications that manage their own memory can get negatively affected with memory reclamation techniques used in overcommitted scenarios in virtualization.When conventional ballooning is used for memory reclamation then application's view of memory allocated to it gets disturbed and being unaware of this disturbance its memory management decisions are badly impacted. The paper [11] tries to give solution for this problem.

- Hardware support for MMU virtualization proves advantageous as:

  - This technique does away with intervening of the VMM whenever there is a updation of VM page table.
  - It also removes the involvement of VMM for VM context switches as VM can change the page table root on its own.

  It also have disadvantage:
  Whenever there is a TLB miss ,there is a requirement of 2D page table walk.

  Exploiting the benefits of hardware support by using host large pages(2MB).If host has sufficiently large memory then this large page support for guest small page will be quite useful but if memory is limited and there is overcommitted situation , host is required to reclaim memory.Now the least expensive technique of memory reclamation technique is memory sharing which comes with little overhead.But there is inherent problem with large pages and page sharing as there is very less probability of content of 2 large pages to be same.Sharing is used as deferred approach by breaking large pages but if memory pressure builds fast then host has to use ballooning and in worst case host level page swapping.Thus this degrades performance. The paper [8] tries to solve this problem.

- CMD: Classification-based Memory Deduplication through Page Access Characteristics

Memory deduplication based on content of pages is a memory reclamation technique in overcommitted situations. Content based page sharing is a technique which can be used for memory deduplication.KSM(Kernel same-page merging) is a technique which uses content based page sharing but problem with this technique is unnecessary comparisons of a candidate page if the content doesn't match with any page.A lot of CPU cycles is wasted for these comparisons and therefore performance degrades. The paper [4] tries to solve this problem.

4.3 Performance Implication of Virtualization

The problem in this category is:

- Understanding Performance Implications Of Nested File Systems in a Virtualized Environment

With virtualization as the VMs are given the illusion of physical resources with virtual resources , there is a need of layer of abstraction for handling resources.The layer of abstraction complicate the understanding ,analysis and management of virtualized environment. The direct impact of this abstraction is nested file system i.e.guest file system over host file system. Consider a dedicated system where there is only one file system for storage media ,therefore real block allocation and file placement decisions are done once with no conflict and we get good performance. Now consider VMs running over host.Each VM is having its own file system which works independently completely unaware that it is running over another file system which is handled by host.There can be conflict in the decisions of both the file systems which would impact the performance of VMs.

4.4 Application and Usecases due to Virtualization

The problems in this category are:

- Improving Remote Desktopping through Adaptive Record/Replay
  Remote desktopping is a technique to access a computer remotely.It can prove useful in many situations such as:
  - Personal use where a user may access office computer from home or in collaborations where multiple users share the same screen.
  - For remote technical support where support staff access the user's computer to rectify the problem.
  - For security check on user machines.

  The remote desktopping tools works by continuously streaming the display from remote server to the client's computer.The bandwidth requirement for this display streaming becomes bottleneck.One possible solution can be the use of compression algorithms for compression of stream data so that bandwidth usage can be reduced.But running these algorithms significant CPU usage is required. Therefore the challenge for this problem is to minimize bandwidth requirement.

5. Problems in memory management

5.1 Memory Management

Virtualization comes with the benefit of running several VMs on a single machine. Physical memory allocations to various VMs can be done in two ways: 1.Static Allocations Each VM is given a chunk of physical memory statically. 2.Dynamic Allocations The memory actually present with the VM can be different from its perception of its available memory. Static allocation comes with the disadvantage of under-utilization of memory since a VM won't be using all its memory all the time. Therefore this leads to inefficient utilization of memory and hinders the primary benefit of virtualization. But this allocation is easy to implement with less complex management policies.

Physical server which is provided with advance hardware support is underutilized when used as dedicated machine for a single user. This under utilization or inefficient utilization of resources motivated the use of virtualization technology. Thus came server consolidation which is the primary benefit of virtualization used in data centers and cloud environment. Suppose with server consolidation 4 VMs are packed in one single physical server and they will share the resources among them.Let us specifically look at how is physical memory P is shared among the VMs and each is given a illusion of having physical memory as p1 ,p2,p3 and p4 respectively.Server consolidation claims to efficiently utilize the physical memory therefore it overcommits the physical memory which means:

$$p1 + p2 + p3 + p4 > P$$

Now certainly this situation needs dynamically handling the memory of different VMs in the server.There is a need to even reclaim memory when required from VMs.Therefore some controlling policy is required as from which VM should the memory be reclaimed and which page to be reclaimed.Such decision making makes dynamic allocation complex.

These high-level management policies require lower-level different techniques of memory reclamation.There are various techniques for memory reclamation such as:

- Ballooning
- Page Sharing
- Host level page swapping

These techniques can behave differently in different situations.Sometime one is better than the other and at other times we may want to use the technique in a different way for getting a better performance in a particular situation.There could even disadvantage of these techniques.

Further chapters will elaborate upon these different situations and disadvantages.Problems in such situations and solutions possible.

6. Memory management with Large page backing for VMs

Hardware advancement has added a feature in the hardware that would make it easier for VMM to run virtual machines. In this particular section we will look upon the hardware support for MMU virtualization.

6.1 Non Virtualized vs Virtualized MMU

The MMU comprises of two structures:

6.1.1 Non-virtualized

- Page table walker
- Translation look aside buffer(TLB)

CPU accesses memory with virtual address and whenever there is such request ,segmentation hardware converts virtual address to linear address by adding segment base. Then this linear address is used by page table walker for traversing the page table tree for getting the physical address(PA). This pair (LA,PA) is inserted into the TLB.TLB is to make for minimizing future addressing overhead.

6.1.2 Virtualized MMU

In virtualized scenario it is the task of the VMM to virtualize MMU. With virtual machines we can even see virtualization of virtual memory in VMs. Therefore VMM has to do some extra work by remapping the addresses a second time from physical address given by VM to machine address.

6.2 Hardware support for MMU virtualization

When there is a hardware support for MMU virtualization then the two levels of address mapping is performed in hardware. The physical MMU points to 2 page-tables.

- The first is the page table maintained by the VM containing VA− >PA mapping.
- The second is the page table maintained by the VMM containing the PA− >MA.

Then the page table walker can walk these two page tables and get the physical address for the virtual address. This LA− >PA can be then cached into the TLB.

6.2.1 Advantages

This technique does away with intervening of the VMM whenever there is a updation of VM page table. It also removes the involvement of VMM for VM context switches as VM can change the page table root on its own.

6.2.2 Disadvantage

In this case of hardware support for MMU virtualization whenever there is TLB miss it can cost more as it must be serviced with 2 level page-table walk.

6.3 Large page backing for virtual machines

As we have already seen hardware support for MMU has several advantages therefore we can exploit them to get better performance in virtualized scenarios. But there was disadvantage which should be kept in mind.For that purpose concept of large page backing comes into play because with large pages there will be less possibility of TLB miss and performance improves. Server consolidation implies overcommitted situation.And if we use host large pages then there are two possibilities:

- If the host has large amount of memory then host large pages will be of lot of use in overcommitted situation.
- If the host memory is limited then in overcommitted situations host large pages will create memory pressure on the host quite early and host is requires to reclaim memory from the VMs.

6.3.1 Problem

There is an inherent problem with host large pages and page sharing between VMs as with a large page there is a very less probability that page will share content with some other page.Therefore this restricts the page sharing based memory reclamation and host has to resort to more expensive memory reclamation techniques like ballooning and host swapping.It even uses page sharing but that is a deferred mechanism by breaking the large pages.This leads to sub optimal performance as as memory pressure grows on the host it will start using ballooning and host swapping this will negate the effect of using page sharing.Both ballooning and host swapping requires swapping of pages which requires a disk access and increase latency.

6.3.2 Challenge

The challenge in the above problem is largely using page sharing as memory reclamation technique while still having the benefits of host large pages.

6.3.3 Solution

Basic objectives of the solution:

- Preserving large page facility if host is under no memory pressure.
- Reclamation of memory should be mostly through page sharing.Therefore focus should be to avoid the situation where the only solution left is reclamation through ballooning or host swapping.

Basic policy for the solution:

- Proactively breaking the pages which are infrequently used.This would reduce the memory pressure.
- If memory pressure is still there then after a certain threshold start breaking even the pages which are frequently used.
- If there is no sharing opportunity in a large page then preserve it.
- If memory pressure reduces then adjust the threshold so as to avoid unnecessary breaking of pages.

7. Applications that manage their own memory running in VMs

There are various applications which manage their own memory for efficient working.Their decision making depends on the actual picture of the memory they are having.These applications are aware of their usage behaviour and use memory accordingly so that they perform better.For example in a database application in a banking scenario.The application knows that some event(query) is going to occur at some fixed time of the day , therefore if it has actual picture of memory then it can make use of that memory for such event so that it can optimize its performance.

7.1 Problem

Having said about these applications ,what happens when such applications run in VMs.Server consolidation leads to multiple VMs packed in a single physical server.
Overcommittment of memory is a primary benefit with server consolidation.Thus dynamic management of memory resources is required.As already discussed in memory management the basic requirement is how to reclaim memory from VMs so that memory can be shared efficiently. The low level techniques for memory reclamation are already discussed and one of them is ballooning.Suppose memory is reclaimed through ballooning from VMs running applications which are managing their own memory. This ballooning can create problem with such applications.We can understand with the help of the example given in the introduction of the chapter.In that example it was said that application has actual picture of memory and also knows the behaviour of the events that can occur.On the basis of which it makes some decisions e.g. it want to keep a page at a particular time of the day irrespective of its usage at other times of day.But what happens when ballooning reclaims memory and it can reclaim the same page based on its usage characterstics and application is not aware of such reclamation.Now the page is swapped to disk and when application tries to use that page ,its request is required to go to the disk and getting that page from there.This can adversely increase the response time of that application and degrades its performance.

7.2 Challenge

- Allocating RAM to applications which manage their own memory such that the applications are not reconfigured or shut down.
- Preserving the ability of an application to optimize performance based on having an accurate idea of the available physical RAM.
- minimal changes to the application.

7.3 Solution

Basic idea of the solution: Extending the existing OS-level ballooning such that it can create an end-to-end solution for memory reallocation ,coordinated across all the three software layers: VMM , OS and application.

The solution is implemented as Application level ballooning.The basic working of ALB involves following steps:

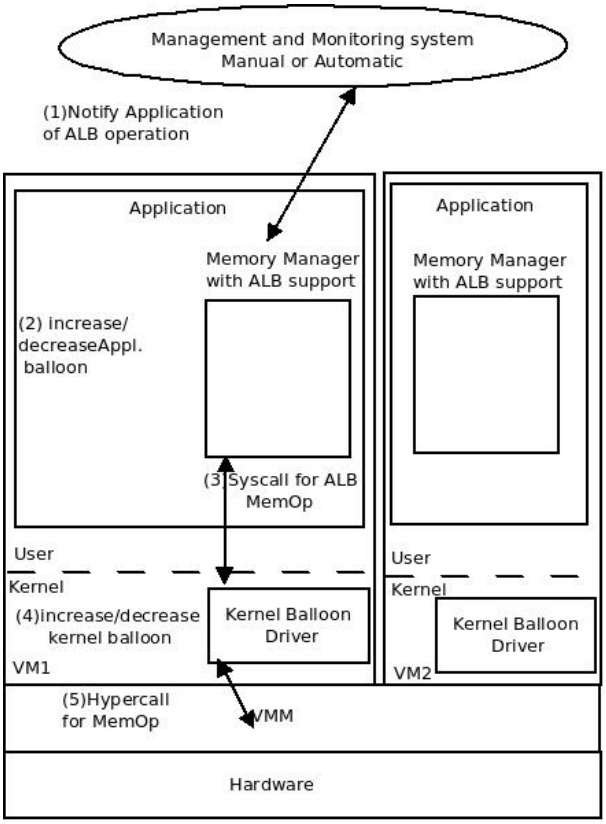

Figure 7.1: ALB functioning

- A management and monitoring system determines policy and conveys a target bal- loon size to the ALB module of the ALB aware application via an RPC call to a local socket.
- The application allocates pages.
- Notifies the OS via a system call.
- The modified guest OS balloon driver processes the requests
- Makes the memory changes visible to Xen with a hypercall.

8. Classification based Memory deduplication

Memory deduplication is one technique for memory reclamation in overcommitted physical server.In this technique only one copy of a page is maintained among the VMs if the content is same.This would help host to reclaim memory.For this host needs to try matching the contents of pages of VMs. Doing all this should keep in mind the trade-off that efficient utilization overhead should not negate its benefit.Kernel same page merging(KSM) is one such method for memory deduplication.It is based on content based page sharing.

8.1 Problem

KSM works by scanning pages of the physical memory.The scanning of pages is iterative.It maintains two global red-black trees which are indexed with the content of pages:

- Stable tree: It maintains search structure for shared pages.In each scan round the page is first compared with the pages in stable tree and if a match is found then that page will be merged with the matched KSM page.
- Unstable tree: This tree maintains the pages which are not shared.Whenever there is a miss in the stable tree, KSM searches this tree and if found then the page is removed from unstable and merged with the candidate page.Then it is shifted to the stable tree.If no match found even in this tree then the page is added to the unstable tree.

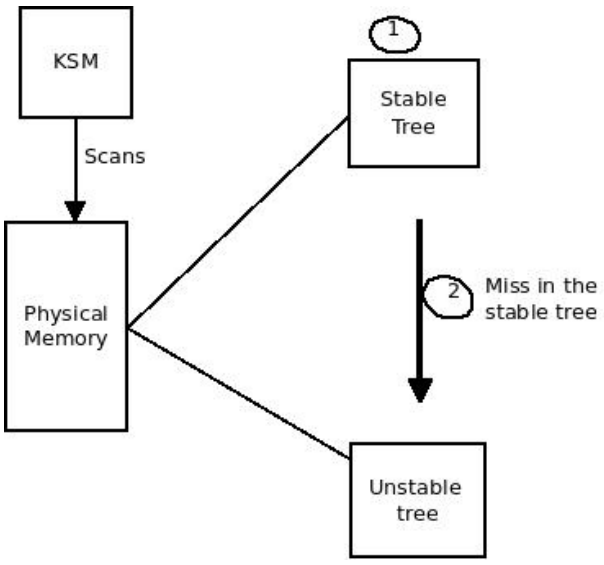

Figure 8.1: KSM working

These trees grow in size with further scanning.As there is no intelligence involved in searching for the pages therefore candidate pages needed to be compared with lot of uncorrelated pages.These comparisons will increase proportionally.This would unnecessarily increase the CPU overhead.

8.1.1 Experiment for establishing problem

QUESTION:How much time of the KSM run time is spent in unnecessary comparisons?? ANSWER:Out of the 44% which is spent on comparison , 83% is spent on unnecessary comparisons.

8.2 Solution

Basic Idea of the solution: Breaking those big trees into small trees intelligently such that a candidate page is compared with only those pages which have high probability of having same content as that page. This would certainly save from unnecessary comparisons and decrease CPU overhead.

There is challenge as to what criterion should be chosen for dividing the pages into smaller trees.This is done by grouping the pages with same access characterstics in same classification.
2 things are required for page classification:

- Pages with high probability to have the same content should be grouped together.
- The classifications should be balanced.This means that the nodes present in each classification should be nearly equal.

There can be three possible page classification algorithms:

- CMD Address

  In this algorithm pages are classified by their physical addresses.For example for 8GB memory setup we can have 8 classifications of 1GB.This would certainly reduce comparisons in each classification but sharing opportunities will be lost because it does not take access characterstics into account.

- CMD Page Count:

  Pages are classified according to their write count.Range of page count is defined and pages with write count in that range would be classified in the same classification.Thus with access characterstic consideration it could result in detection of more page sharing opportunities.

- CMD Sub page Distribution:

  This a more fine-grained approach where a page is divided into sub-pages and access characterstics of all the sub-pages is maintained. Pages with same sub-page access distribution are grouped into same classification.This can more efficiently detect page sharing opportunities.

8.2.1 Solution Algorithm

After the pages are grouped into classifications ,there exist stable and unstable tree for each classification.Comparisons and searching takes place classification-by-classification.Following flowchart will give the KSM tree algorithm for this classification based grouping of pages.

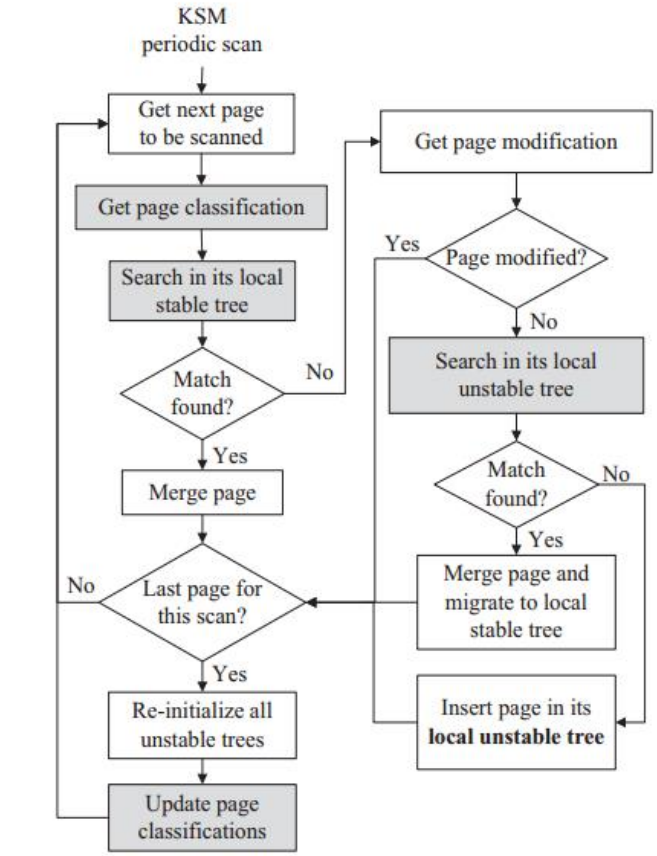

Figure 8.2: Classification based KSM Tree algorithm flowchart

9. *Conclusion*

Categorization of the recent problems gives the importance of each category in the field of virtualization and also points out the challenges which are faced and scope of improvement in the existing solutions. Solutions to problems in memory management category where different variations of memory reclamation were suggested to address the issues with existing solutions.And also including the decision making over various memory reclamation technique so as to incur least cost and maximum performance.

Bibliography

[1]

[2] Nadav Amit, Muli Ben-Yehuda, Dan Tsafrir, and Assaf Schuster. viommu: efficient iommu emulation. In *USENIX Annual Technical Conference (ATC)*, pages 73–86, 2011.

[3] Kapil Arya, Yury Baskakov, and Alex Garthwaite. Tesseract: reconciling guest i/o and hypervisor swapping in a vm. In *ACM SIGPLAN Notices*, volume 49, pages 15–28. ACM, 2014.

[4] Licheng Chen, Zhipeng Wei, Zehan Cui, Mingyu Chen, Haiyang Pan, and Yungang Bao. Cmd: classification-based memory deduplication through page access characteristics. In *ACM SIGPLAN Notices*, volume 49, pages 65–76. ACM, 2014.

[5] Kallol Dey, Debadatta Mishra, and Purushottam Kulkarni. Vagabond: Dynamic network endpoint reconfiguration in virtualized environments. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–13. ACM, 2014.

[6] Sahan Gamage, Cong Xu, Ramana Rao Kompella, and Dongyan Xu. vpipe: Piped i/o offloading for efficient data movement in virtualized clouds. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–13. ACM, 2014.

[7] Jayneel Gandhi, Arkaprava Basu, Mark D Hill, and Michael M Swift. Efficient memory virtualization: Reducing dimensionality of nested page walks. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 178–189. IEEE Computer Society, 2014.

[8] Fei Guo, Seongbeom Kim, Yury Baskakov, and Ishan Banerjee. Proactively breaking large pages to improve memory overcommitment performance in vmware esxi. In *Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 39–51. ACM, 2015.

[9] Ali José Mashtizadeh, Min Cai, Gabriel Tarasuk-Levin, Ricardo Koller, Tal Garfinkel, and Sreekanth Setty. Xvmotion: Unified virtual machine migration over long distance. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 97–108, 2014.

[10] Eunbyung Park, Bernhard Egger, and Jaejin Lee. Fast and space-efficient virtual machine checkpointing. In *ACM SIGPLAN Notices*, volume 46, pages 75–86. ACM, 2011.

[11] Tudor-Ioan Salomie, Gustavo Alonso, Timothy Roscoe, and Kevin Elphinstone. Application level ballooning for efficient server consolidation. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 337–350. ACM, 2013.

[12] Yannis Sfakianakis, Stelios Mavridis, Anastasios Papagiannis, Spyridon Papageorgiou, Markos Fountoulakis, Manolis Marazakis, and Angelos Bilas. Vanguard: Increasing server efficiency via workload isolation in the storage i/o path. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–13. ACM, 2014.

[13] Cong Xu, Sahan Gamage, Hui Lu, Ramana Rao Kompella, and Dongyan Xu. vturbo: Accelerating virtual machine i/o processing using designated turbo-sliced core. In *USENIX Annual Technical Conference*, pages 243–254, 2013.

[14] Yuehai Xu and Song Jiang. A scheduling framework that makes any disk schedulers non-work-conserving solely based on request characteristics. In *FAST*, pages 119–132, 2011.

[15] Irene Zhang, Alex Garthwaite, Yury Baskakov, and Kenneth C Barr. Fast restore of checkpointed memory using working set estimation. In *ACM SIGPLAN Notices*, volume 46, pages 87–98. ACM, 2011.