Midterm Report

Name：Yitong WANG(王奕童) 11910104@mail.sustech.edu.cn

SID：11910104

Lab：Friday5-6节

Lab Teacher：Yun SHEN(沈昀) sheny@mail.sustech.edu.cn

Lab SA:

- Yining TANG(汤怡宁) 11811237@mail.sustech.edu.cn
- Yushan WANG(王宇杉) 11813002@mail.sustech.edu.cn

1. Terminologies of virtualization

1.1 Virtualization definition

Virtualization is used to split the hardware resource of computers so that difference programs can have an isolated running environment and be executed at the same time.
Virtualization allows the software to use the memory abstraction so that it does not to operate on the hardware directly.
Moreover, with virtualization, users can run different programs and applications on the same machine.

1.2 Three usage models of virtualization

Workload isolation

This means in order to tolerate the system fault, the virtualization application first copies an identical version of itself, then runs them in two different VMs with the same workload. Thus the system security could be improved.

Workload consolidation

Work consolidation allows the companies to execute several different programs which requires to be run certain operating systems on several hardware platforms previously, on only one hardware

platform. Work consolidation applied virtualization so that the several kinds of operating systems(older and newer version) at one machine at the same time.

Workload migration

This concept means the operation that moves one running workload from one architecture environment to another.
Saving the application running status in a virtual machine will be a nice choice since it will not depend on the hardware support.
Moreover, the work balancing and failure prediction could improve the stability of migration process.

The best-suited workload for migration:

- capacity varies a lot
- data storage and protection
- micro-service application with several layers
- need to be expanded by the requirement

1.3 Three well-known VMMs

XenoLinux(X), VMware workstation 3.2(V) and User-Mode Linux(U)

1.4 Five concepts

Paravirtualization

Paravirtualization is a kind of hardware virtualization implementation technique.
Paravirtualization uses VMMs to communicate with the local hardware.
It accelerates the program, and does not need to re-compile or cause system fault, or make any changes to the users' applications.
Therefore it needs to have some modification to the users' operating system to fit the corresponding hardware interfaces.
But it cannot support all operating systems. For instance, Xen cannot run on the Windows 10 platform.

Full virtualization

In the full virtualization is another kind of hardware virtualization implementation technique.
This technique uses virtual machine to handle the cooperation between OS and local hardware.
Full virtualization can be faster than the hardware simulation, but slower than the original machine.
And full virtualization does not need to make change to the users' OS.
It can also support several OSes, which needs to support the local hardware.

Binary translation

This is used to translate the original binary code to simulate another instruction set. It can be implemented by hardware or software.
It could be used in the translation of VMMs so that VMMs can support a wider range of target opearting systems.

Hardware-assisted virtualization

Hardware-assisted virtualization is a native implementation of virtualization.
Hardware provides structure to support the virtual machine to allow and watch the user OS to run independently.

Moreover, there are several platform supporting this technique, such as AMD-V and Intel VT-x in x86, VT-d in IOMMU etc.

Hybrid virtualization

Hybrid virtualization is becoming prevailing amont the users.
It can combine several kinds of virtualization technique and gather their benefits.

It has several advantages among the previous virtualization:

1. It can reduce the expense for changing codes in users' applications;
2. It does not spend too much time on the shutdown of operating system;
3. It could be more cheap and cost less money by the developers;
4. It can support several resource access when the resource is required;

Thus it is becoming popular among several companies and actual development environment.

2. Privilege levels

2.1 Privilege

There are 4 privilege levels, whose level number means: the lower level number, the higher the privilege levels.

1. Level 0: OS Kernel
2. Level 1: OS Services
3. Level 2: OS Services
4. Level 3: User Applications

Three examples:

- Example 1: Privilege separation could split one single program to several programs with smaller parts. Thus these programs can only communicate with others via the operating system. This increases the security of the whole system since it cannot attack the part with higher privilege but only cause a denial-of-service attack.
- Example 2: Network applications needs to do some operations with certain privilege. Thus those softwares will give back and separate the privilege, and set the user into a lower-privilege level(i.e. deleting root in Unix). Privilege separation contributes a lot to the permission control.
- Example 3: Privilege separation could be implemented as splitting the program into two processes. The major one does not have privilege while smaller one does. Thus those processes can do the privileged operations but the attack to the major program cannot get the high privilege.

2.2 Ring compression

As described previously, there are 4 levels in the privilege rings.
In order to protect VMM which is from user software, IA-32 uses 2 mechanisms named segment limits and paging.

However, IA-32 cannot differentiate the privilege leve of 0-2.
Thus the guest operating system can only be executed on level 3 and cannot get protection from those applications.
This is called ring compression.

2.3 Ring compression for X86 (IA-32)

IA-32 uses segment limits and paging to handle ring compression in the X86 (IA-32) architecture.

2.4 Ring compression for x86-64

x86-64 uses paging to handle ring compression in the X86-64 architecture.

2.5 Ring aliasing

This appears when a software is executed under the privilege level. But the software is not written in this privilege level.

2.6 VMX root and VMX non-root in VT-x

VMX root: the codes are executed between the instructions `vm-exit` and `VMRESUME`.

VMX non-root: the codes are executed in the normal kernel mode.

2.7 Address the challenge

Some commands can be executed by the virtual machine normal, thus it is not required to simulate the privileged commands.
VMMs are allowed to execute the user software in its prefered privilege level. VMMs are able to be freed from the priviledged simulation commands.

3. System calls, interrupts and exceptions

3.1 Purpose & Difference

Purpose of system call: request for the system service provided by the hardware. OS banned the user to use hardware resource directly, thus the user can only use the interfaces provided by the operating system to access the resource.

Difference:

- System call: user program request the operating system to get the support by the hearware resource.
- Function call: just a program request another one to complete the specific funtion.

3.2 Hypercall in Xen

Hypercall is a synchronous procedure that domain uses to do some operations needing privleges such as updating the pagetables.

3.3 Xen & exceptions

Xen can do the virtualization of exceptions straightforwardly. And the methods are like the virtualization to memory fault, or the trap of softwares.

3.4 Challenges of virtualizing interrupts

Currently, IA-32 uses IF(interrupt flag) in EFLAGS to control the interrupt shield.
And VMMs may manage the external interrupt and prevent the guest softeware to control the interrupt.
And current protection causes fault when guest software try to control the interrupt when the ring deprivileging is performed.

But this method may be failed because some OSs are likely to mask and unmask the interrupts of themselves, which brings a negative effect the the system performance.

3.5 Xen & interrupts

VT-x and VT-i gives explicit support for the interrupts virtualization.

- External-interrupt exiting: VM-execution control. If the control value is 1, the VMM will ban any operation of guest trying to modify EFLAGS.IF. VT-i has virtualization-acceleration. It can ban the guest software to affect interrupt masking and ban transition to VMM.
- Interrupt-window exiting: VM-execution control. If the control value is 1, the VM exit will be triggered, when the guest software process is receivint interrupt signal. VT-i uses PAL service to register the pending of the virtual interrupt.

Benefit: This could make the virtualization of interrupts safer and faster. Thus the virtualization will be of more safety.

3.6 VMCS

VMCS

VMCS can manage the entries and exits of VMs, and the behaviour of processor in the operations of VMX non-root.
VMCS can implement the virtualization of CPUR in Intel x86 and record the vCPU status.

VM exit & VM entry

- VM exit: the transition from current running VM to VMM, since VMM must gain the system control for some reasons. This operation lets CPU change into VMX Root status.
- VM entry: the transition from VMM to current running VM. This operation lets CPU change into VMX non-Root status.

How are VMCS used

VMCS has a state area for the guest and host, and this area can be saved at the VM entry and VM exit.
And VMCS which can control the guest operation selection will cause VM exits.

3.7 Virtualize interrupts by Xen and Intel VT-x

Intel VT-x is used in the storage of Xen. This provides the support of "virtual processor" abstraction to the guest operating system.
Thus the "virtual procressor" could contribute to the virtualiztion of interrupts and make the virtualization easier.
But Xen take the responsibility of managing devices, and enforcing the resource isolation.

3.8 Intel VT-x support for exception virtualization

Exceptions and other instructions are likely to cause VM exits conditionally. This needs the support of Intel VT-x and uses the VM-execution control fields, which is a part of VMCS.

4. Address translation

4.1 x86 (IA-32) address translation

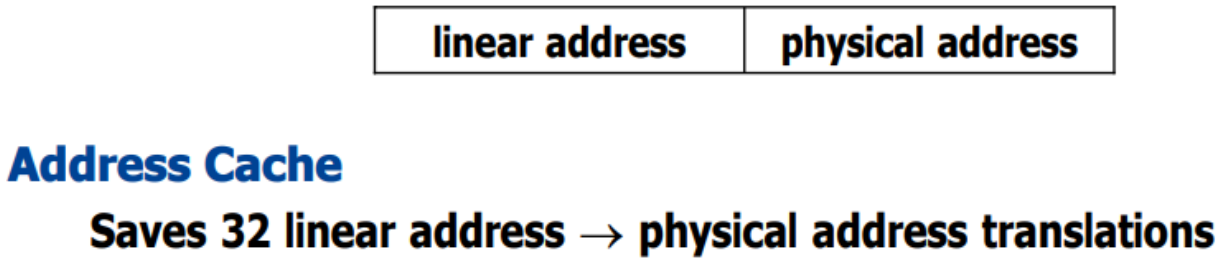
The translation is mapping the logical address to the physical address. It has several steps:

1. Logical Address -> Linear Address, via segmentation unit
2. Linear Address -> Physical Address, via paging unit

The first procedure is done by:

```
Linear Address = Base Address + offset.
Base Address = Linear address of first byte
```

The second procedure is done by the page table, which stores the mapping from 32-bit linear address to 32-bit physical address.

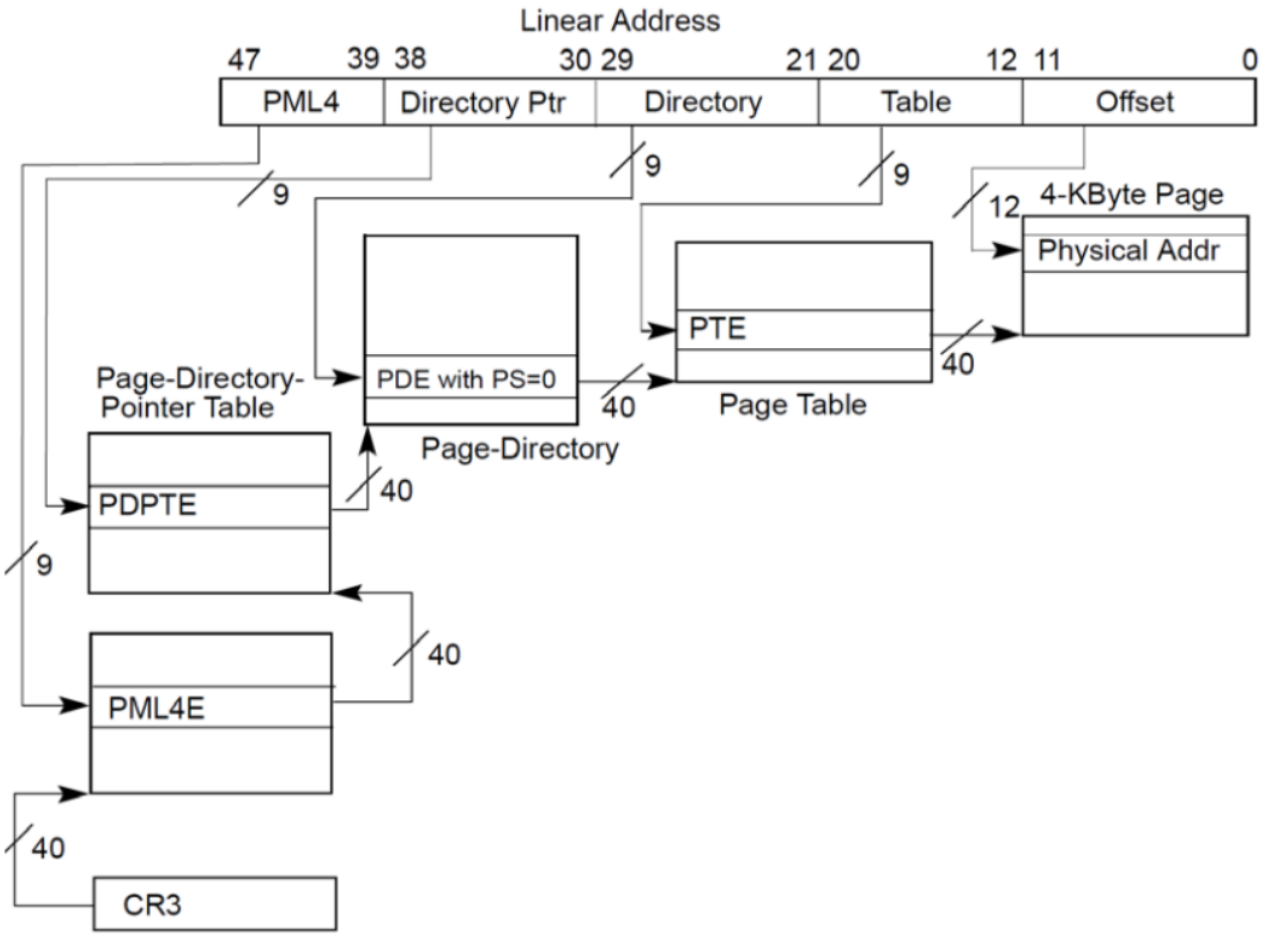


> reference: a part of slide in https://cs.hac.ac.il/staff/martin/Micro_Modern/slide03.pdf

4.2 x86-64 address translation

x86-64 also uses page table lookup to do the mapping from logical address to physical address. Moreover, in the x86-64 design, the page table is widely implemented by multi-level structure. This is because multi-level design can save memory for the page table. The page table we try to access may not have a valid physical address.
Therefore, the tree of multi-level page-table may be sparse thus the memory could be saved.

The following figure from https://blog.csdn.net/hhhanpan/article/details/80548687 shows the translation within x64-64.

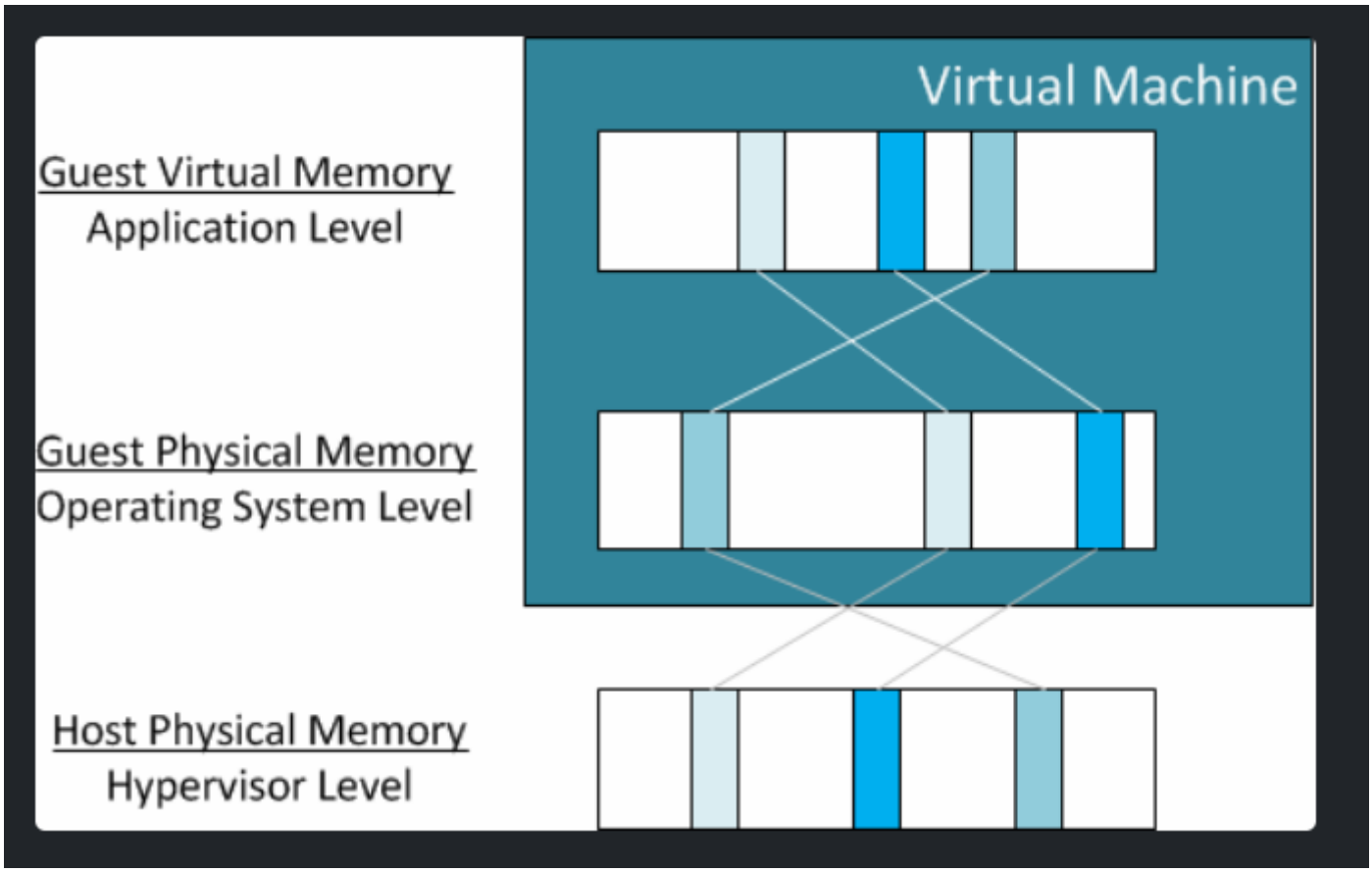


4.3 Memory relation

- Guest virtual memory: This is the continuous virtual address space provided by the guest OS. It is visible to the running applications in the virtual machine.
- Guest physical memory: This is the physical memory provided by the host physical memory. This helps VMMs to give the mapping from the memory of gusest to it of host.
- Machine memory: Besides virual memory and physical memory, the hypervisor adds an additional layer, which is named Machine Memory. This part is visible to the hypervisor on the system.

Their relation:

1. Virtual memory provides uniform address space.
2. The virtual address is mapped into physical memory by the guest operating system.
3. The physical memory is mapped into machine memory by the hypervisors.



> Disgram reference: https://www.simongreaves.co.uk/vmware-virtual-machine-memory-guide/

4.4 Xen management

OS will do the mapping, from virtual addresses to the physical ones.
And VMM will do the mapping to locate the machine addresses through the pre-OS page tables.

4.5 Address-space compression

The guests operating systems may required to access the whole virtual memory space of the processor, and there are two points to be noticed:

- VMM can run in the virtual address space of guest, but it also occupy large amounts of virtual address space.
- VMM can run in the separate address space. But it needs to save virtual address space to manage the transformation between guest software and VMMs.

In the procedure, VMM will ban the guest to use the parts of using those address space being used, which is called address-space compression.

4.6 Xen solution to 4.5

Xen solve this problem using VT-x.

1. When the system is doing the transition between user software and VMMs, the linear address space is changed thus allows the guest software to make full use of the address space.
2. VMCS will manage the transitions of VMX. This resides in the physical address space, but not on the linear one.

> reference: http://pages.di.unipi.it/tonelli/baiardi/didattica/SR/2016/2.0-virtualization.pdf

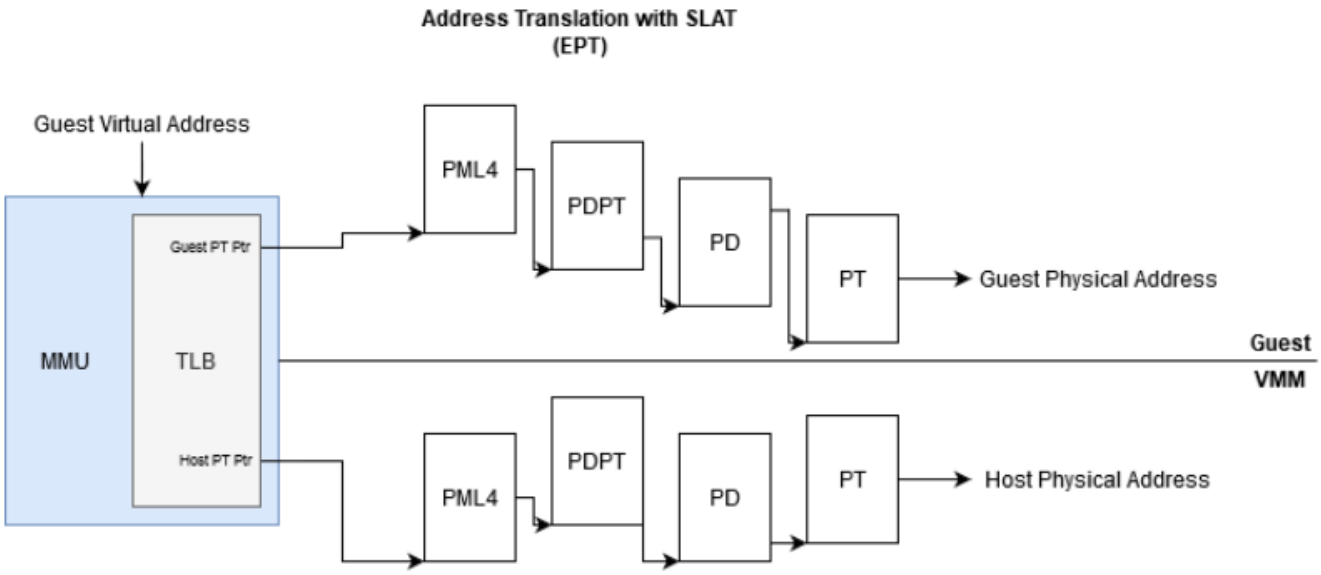
4.7 Intel EPT

Intel EPT is an implementation of SLAT. It was firstly introduced in the Nehalem microarchitecture, which could be found in several processors such as Core i7, i5 and i3.

Intel EPT is release as second generation of hardware support, which contributes to the MMU virtualization.
In the virtualization with SLAT, a virtual address is passed into MMU and TLB do a look-up to check the existence of the mapping between virtual address and physical address.

- Mapping exist. We get TLB hit and get the address from the TLB section.
- Mapping miss. We need to traverse the multi-level page structure to get the translation result.

The following figure shows the procedure:



> reference: https://revers.engineering/mmu-ept-technical-details/

4.8 Xen allocation

When the domain is created, the initial memory is allocated to support strong isolation.
But the domain may require different amount of memory resources.

- Case 1: The domain needs more memory resource. It will request more momory page from Xen, until it reaches the reservation limit.
- Case 2: The domain does not need so much memory. It will free the memory page back into Xen to save the memory.