



Maintaining cache coherency

Disheng Su (Disheng.su@intel.com)

Yunhong Jiang

Eddie (Yaozu) Dong



Agenda

Cache coherency challenges

Background: Intel® 64 and IA-32 Cache control

Virtualization for cache coherency

- **Emulating cache mode**
- **Emulating EMT (Effective Memory Type)**

Future work

Backup



Software and Solutions Group



Challenge of Cache coherency

Cache is processor internal, and invisible to devices

Pass through devices expect to see correct operation from driver with correct ordering

- **MMIO R/W**
 - Caching of MMIO leads to incorrect device behavior directly
- **Shared RAM**
 - Incorrect caching of RAM, such as CMD buffer in video, leads to device malfunction too



Challenge: Emulating cache mode

Different cache mode has different cache behavior

- **CRO.NW & CRO.CD decide the cache mode: Normal or no-fill**

Host and guest are sharing cache in Xen

- **Host typically works in highest performance mode, i.e. normal cache mode**
- **Guest OS uses no-fill mode before changing MTRR/PAT MSRs**

Hardware doesn't support cache mode switch at VM Exit/Entry time

- **CRO.NW & CRO.CD bits are ignored at VM Entry**
- **Even hardware mode switch is supported, hypervisor still can't guarantee identical cache contents for guest no-fill mode**



Agenda

Cache coherency challenges

Background: Intel® 64 and IA-32 Cache control

Virtualization for cache coherency

- **Emulating cache mode**
- **Emulating EMT (Effective Memory Type)**

Future work

Backup



Software and Solutions Group



Background: Cache Operating Modes

CR0.CD/CR0.NW	Caching and Read/Write Policy
0/0	Normal Cache Mode , Highest performance cache operation (External snoop traffic is supported)
0/1	Invalidate setting
1/0	No-fill Cache Mode . Memory Coherency is maintained (External snoop traffic is supported) <ul style="list-style-type: none">•State after a processor power up or reset•Read hits access cache; read misses don't cause replacement•Write hits update cache; write misses access memory•Strict memory ordering is not enforced unless the MTRRs are disabled and/or all memory is referenced as uncached.
1/1	Not supported in P4 and more recent processor families, and select no-fill cache mode.



Background: Memory Types

Memory Type and Mnemonic	Cacheable	Writeback Cacheable	Allows Speculative Reads	Memory Ordering Model
Strong Uncacheable (UC)	No	No	No	Strong Ordering
Uncacheable (UC-)	No	No	No	Strong Ordering. Can only be selected through the PAT. Can be overridden by WC in MTRRs.
Write Combining (WC)	No	No	Yes	Weak Ordering. Available by programming MTRRs or by selecting it through the PAT.
Write Through (WT)	Yes	No	Yes	Speculative Processor Ordering.
Write Back (WB)	Yes	Yes	Yes	Speculative Processor Ordering.
Write Protected (WP)	Yes for reads; no for writes	No	Yes	Speculative Processor Ordering. Available by programming MTRRs.



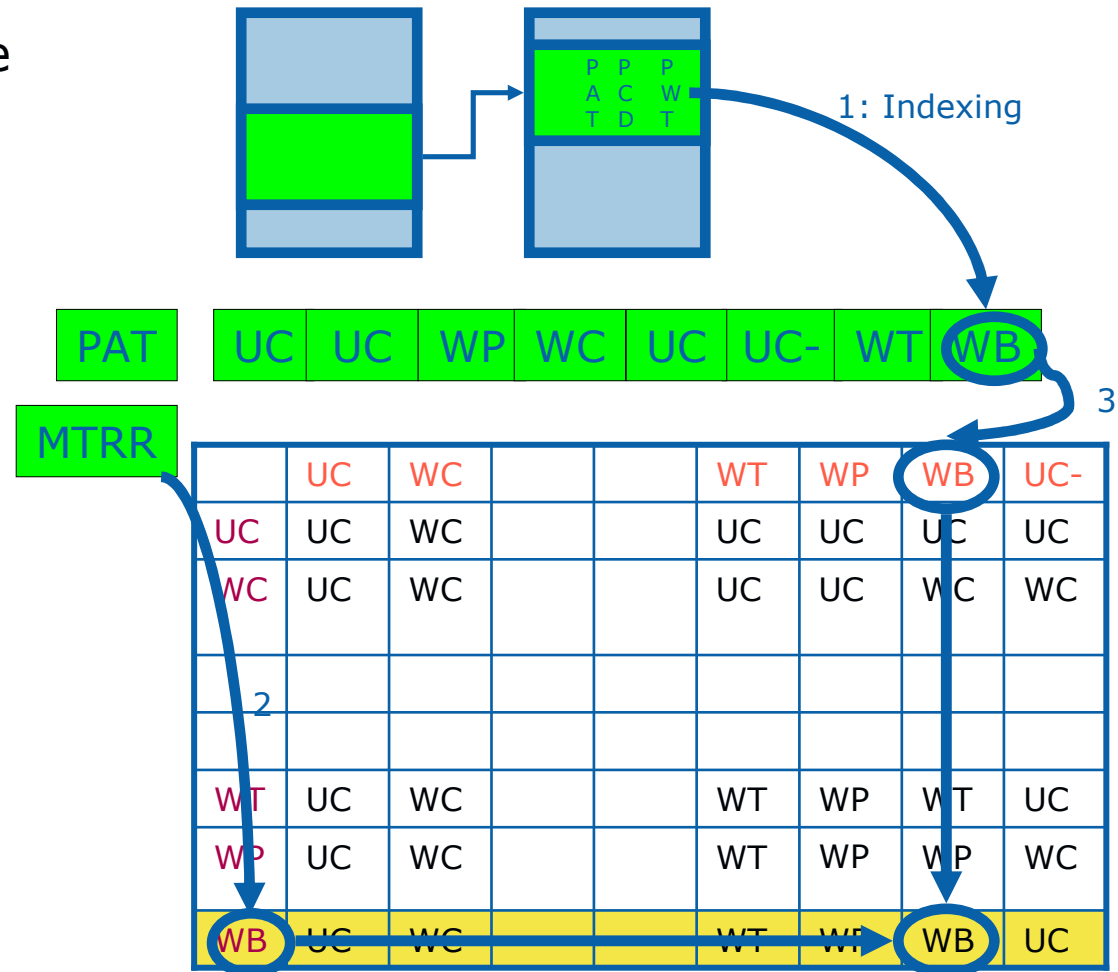
Background: Effective Memory Type (EMT)

PAT/PCD/PWT in page table entries forms an index to PAT Entry

- IA32_CR_PAT (MSR) contains 8 page attribute fields

Memory Type Range Registers (MTRRS)

- Providing a mechanism for associating the memory types with physical address ranges in the system memory



Agenda

Intel® 64 and IA-32 Cache control overview

Cache coherency challenges

Virtualization for cache coherency

- **Emulating cache mode**
- **Emulating EMT (Effective Memory Type)**

Future work

Backup



Software and Solutions Group



Emulating cache mode

Physical processor always runs in normal cache mode

Guest no-fill mode is emulated by forcing UC for whole guest memory

- **WB may work, but ...**
 - Guest enters no-fill mode and invalidates all caches, device drivers may expect to see in order access to memories.
 - Not sure if DOS will work in this way
- **UC is a kind of special no-fill mode behavior where no single cache line is hit**
 - Once one VCPU enters no-fill mode, or virtual MTRR/PAT setting is not identical, forcing all VCPUs to use UC mapping for correctness

A typical guest only enter no-fill mode when MTRR/PAT changed which is non performance critical



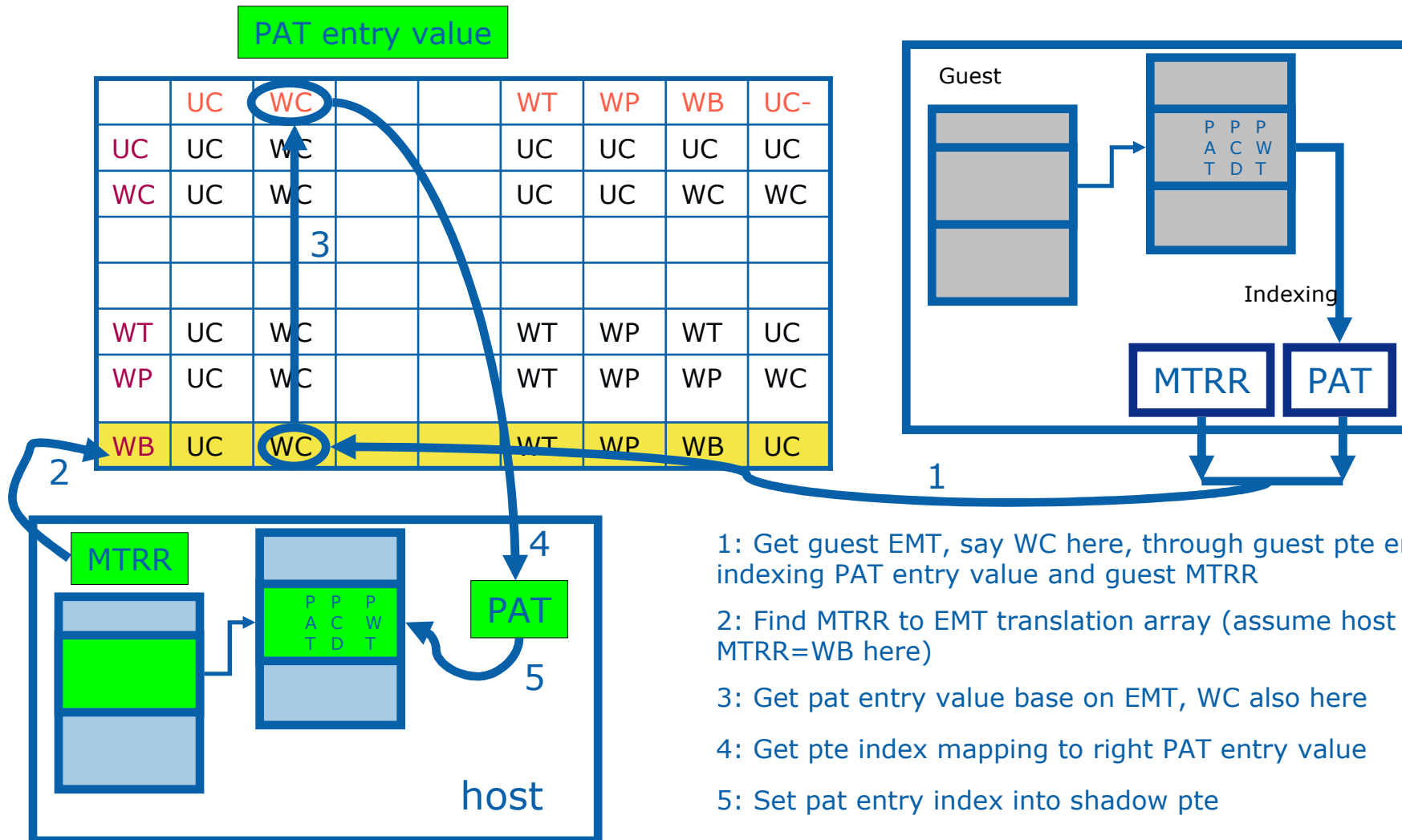
Propagating guest EMT

Guest EMT is propagated by choosing PAT entry in shadow page table (for both MMIO & RAM)

- **Host IA32_CR_PAT is modified at VMX startup time to provide all kind of PAT entry value**
- **Host MTRR is never changed**
 - Non modified host MTRR plus a right chosen PAT entry value to reach effective memory type as guest required
 - Shadow PAT entry finding may be failed, but fine so far 😊
- **Guest MTRR/PAT is virtualized**
 - Shadow page tables are blow out when guest modifies MTRR/PAT or cache mode



Determine shadow PAT entry index



- 1: Get guest EMT, say WC here, through guest pte entry indexing PAT entry value and guest MTRR
- 2: Find MTRR to EMT translation array (assume host MTRR=WB here)
- 3: Get pat entry value base on EMT, WC also here
- 4: Get pte index mapping to right PAT entry value
- 5: Set pat entry index into shadow pte

Agenda

Intel® 64 and IA-32 Cache control overview

Cache coherency challenges

Virtualization for cache coherency

- **Emulating cache mode**
- **Emulating EMT (Effective Memory Type)**

Future work

Backup



Software and Solutions Group



Future work

Enable EPT based guest EMT propagation

Hypervisor needs to keep track of EMT for guest page and refuse to map if it conflicts

- A previous guest page table page may be used as guest DMA buffer, and thus be mapped as UC for example
- Shadow code may be not aware of this changes, and remap it later on using default EMT, i.e. WB → Alias mapping conflict
- Qemu map_cache may map guest page in WB too



Backup



Software and Solutions Group



Avoid different EMT for alias mapping

Propagating of guest effective memory type may cause alias mapping issue if host maps (usually WB) too

- **MMIO of emulated devices usually isn't mapped by host**
- **But, VRAM is an exception**
 - Guest usually set WC
 - Qemu maps it as WB

Solution

- **Pin effective memory type to WB for certain range of memory**
 - A new hypercall is used

Will other host module also map guest page? Such as shadow?



Maintaining explicit cache coherency

Device drivers of pass through devices may not use UC for shared memories

- **Audio driver may map audio buffers as WB for performance, and use WBINVD to flush cache contents before using as DMA buffer**
- **If vcpu migrates, a guest WBINVD instruction can't flush dirty contents in VCPU to memory →lost of cache coherency**

Solution

- **For WBINVD EXIT capable processors**
 - **Execute WBINVD on each dirty processors when VMExit for WBINVD**
- **For WBINVD EXIT incapable processors**
 - **Flush cache at VCPU migration time**



Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.

Intel may make changes to specifications, product descriptions, and plans at any time, without notice.

All dates provided are subject to change without notice.

Intel is a trademark of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2007, Intel Corporation. All rights are protected.



Software and Solutions Group



