

Maintaining cache coherency for pass through devices

Disheng Su
disheng.su@intel.com
Intel Open Source Technology Center

1. Abstract

Maintaining cache coherency is very important for pass through devices in virtualization environment. It not only impacts the performance, but also the correctness of guest. In this presentation, we will talk about the challenges of maintaining cache coherency, and the current status in Xen. Furthermore, we will also talk about the needed future work.

2. Background

In IA32, there are two ways to control cache. The one way is to control cache operation mode: normal cache mode and no-fill cache mode, which specified in the combination of CR0.CD and CR0.NW. In normal cache mode, it has the highest performance cache operation, widely be used. On the other hand, in no-fill cache mode, it has bad performance, due to not fill cache line even when write misses. No-fill cache mode is rarely used, only in such as MTRR/PAT initialization or modification. The other way is the combination of Memory Type Range Register (MTRR) and Page Attribute Table (PAT). MTRR provides a mechanism for associating the memory types with physical address ranges in the system memory, where PAT allows mapping of memory types at pages within the linear address space.

With cache mode control, one can control cache operation, no matter which address is accessed. With MTRR/PAT combination, one can control memory types for a specified address or an address range. This memory types are Strong Uncacheable (UC), Uncacheable (UC-), Write Combining (WC), Write Through (WT), Write Back (WB), Write Protected (WP). UC is usually used for MMIO range, WC is for VRAM and WB is for RAM. Other memory types are rarely used.

3. Challenges

There are challenges to make sure cache coherency in virtualization environment. The first one is cache mode difference between host and guest. Currently, in hardware-assisted virtualization technology, such as VT, host and guest are sharing the same cache. Host typically works in highest performance mode: normal cache mode. And sometimes, guest OS needs to enter no-fill mode, such as during MTRR/PAT initialization. But hardware doesn't support cache mode switch between host and guest. Guest CR0.CD and CR0.NW are ignored at VM Entry. So it means guest is always in the same cache mode as host, even guest wants to change its cache mode to be different from host. Guest will be failed in such case if it takes the assumption on the no-fill cache mode, but it actually is in normal cache mode as host. Moreover, even cache mode switch is supported by hardware, host still can't guarantee identical cache contents for guest, which is a matter of correctness when guest is in no-fill mode.

Another one is guest with pass through devices need to access assigned hardware with proper memory type. In Xen, legacy HVM guest memory and frame buffer are

backed by RAM, which is mapped as WB as default. Things need to be changed for pass through domain. MMIO address range for pass through device need to be set as UC. Even system RAM shared between guest and hardware may need proper effective memory type. Because PCI-E device driver may enable no snoop mechanism, and then map RAM as UC or WC.

We address these challenges in next section.

4. Proposal

We address cache mode issue by forcing UC for whole guest memory when guest in no-fill mode. Physical CPU always runs in normal cache mode in order to get high performance. UC is a kind of special no-fill mode behavior where no single cache line is hit. So we can use UC to emulate no-fill mode. And a typical guest only enters no-fill mode when MTRR/PAT changed, so forcing UC at this time is non performance critical.

Guest effective memory type for pass through domain is addressed by MTRR/PAT virtualization. Guest effective memory type is propagated by choosing PAT entry in shadow page table. Host PAT MSR is modified at VMX startup time to cover all kind of PAT entry value. Non-modified host MTRR plus a right chosen PAT entry value to reach effective memory type as guest wanted.

Accompany with the above proposals, following issues are also needed to be addressed: avoid cache alias mapping and explicit cache coherency maintaining. Propagating of guest effective memory type may cause cache alias mapping if someone (host or other domains) maps the shared range as other memory type. MMIO of emulated devices usually isn't mapped by host with exception of VRAM. Guest wants to set VRAM as WC, but Qemu sets it as WB. So cache alias happens. We propose to pin effective memory type as WB for certain range of memory.

And guest may explicit maintain cache coherency. Device drivers of pass through devices may not use UC for shared memories. For example, 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 at that time, a guest WBINVD instruction can't flush dirty contents in VCPU to memory. So cache coherency can't be maintained as guest wanted. So WBINVD needs to be virtualized by executing WBINVD on each dirty processor when VM Exit for WBINVD on WBINVD EXIT capable processor, or by flushing cache at vcpu migration time.

5. Future work

We will add memory type for EPT. EPT will ease the above development a lot.

Also the hypervisor needs to keep track of guest effective memory type for guest page and refuse to map it if it conflicts with guest.

6. Conclusion

In this paper, we present the challenges in maintaining cache coherency for pass through devices. Then we introduce our proposals to address these challenges. All are done in Xen 3.2. We think that maintain cache coherency will become important with the pass through devices used more widely in the future.

Acknowledgements

We would like to thank Yunhong Jiang, Xiaowei Yang, Kevin Tian and Eddie Dong from Intel Open Source Technology Center for the kind help during the development.