

[PATCH 11/40] KVM: MMU: Decouple mmio from shadow page tables

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-03/msg12017.html>

- *From:* Avi Kivity <avi@xxxxxxxxxxxxx>
 - *Date:* Mon, 31 Mar 2008 17:36:55 +0300
-

Currently an mmio guest pte is encoded in the shadow pagetable as a not-present trapping pte, with the SHADOW_IO_MARK bit set. However nothing is ever done with this information, so maintaining it is a useless complication.

This patch moves the check for mmio to before shadow ptes are instantiated, so the shadow code is never invoked for ptes that reference mmio. The code is simpler, and with future work, can be made to handle mmio concurrently.

Signed-off-by: Avi Kivity <avi@xxxxxxxxxxxxx>

arch/x86/kvm/mmu.c | 34 ++++++-----
arch/x86/kvm/paging_tmpl.h | 17 ++++++-----
2 files changed, 23 insertions(+), 28 deletions(-)

```
diff --git a/arch/x86/kvm/mmu.c b/arch/x86/kvm/mmu.c
index 6f8392d..6651dfa 100644
--- a/arch/x86/kvm/mmu.c
+++ b/arch/x86/kvm/mmu.c
@@ -101,8 +101,6 @@ static int dbg = 1;
#define PT_FIRST_AVAIL_BITS_SHIFT 9
#define PT64_SECOND_AVAIL_BITS_SHIFT 52

-#define PT_SHADOW_IO_MARK (1ULL << PT_FIRST_AVAIL_BITS_SHIFT)
-
#define VALID_PAGE(x) ((x) != INVALID_PAGE)

#define PT64_LEVEL_BITS 9
@@ -200,7 +198,6 @@ static int is_present_pte(unsigned long pte)

static int is_shadow_present_pte(u64 pte)
{
- pte &= ~PT_SHADOW_IO_MARK;
return pte != shadow_trap_nonpresent_pte
&& pte != shadow_notrap_nonpresent_pte;
}
@@ -215,11 +212,6 @@ static int is_dirty_pte(unsigned long pte)
return pte & PT_DIRTY_MASK;
```

```

}

-static int is_io_pte(unsigned long pte)
-{
- return pte & PT_SHADOW_IO_MARK;
-}
-
static int is_rmap_pte(u64 pte)
{
return is_shadow_present_pte(pte);
@@ -538,7 +530,7 @@ static int is_empty_shadow_page(u64 *spt)
u64 *end;

for (pos = spt, end = pos + PAGE_SIZE / sizeof(u64); pos != end; pos++)
- if ((*pos & ~PT_SHADOW_IO_MARK) != shadow_trap_nonpresent_pte) {
+ if (*pos != shadow_trap_nonpresent_pte) {
printk(KERN_ERR "%s: %p %llx\n", __FUNCTION__,
pos, *pos);
return 0;
@@ -926,13 +918,6 @@ static void mmu_set_spte(struct kvm_vcpu *vcpu, u64 *shadow_pte,
if (pte_access & ACC_USER_MASK)
spte |= PT_USER_MASK;

- if (is_error_page(page)) {
- set_shadow_pte(shadow_pte,
- shadow_trap_nonpresent_pte | PT_SHADOW_IO_MARK);
- kvm_release_page_clean(page);
- return;
- }
-
spte |= page_to_phys(page);

if ((pte_access & ACC_WRITE_MASK)
@@ -1002,7 +987,7 @@ static int __nonpaging_map(struct kvm_vcpu *vcpu, gva_t v, int write,
if (level == 1) {
mmu_set_spte(vcpu, &table[index], ACC_ALL, ACC_ALL,
0, write, 1, &pt_write, gfn, page);
- return pt_write || is_io_pte(table[index]);
+ return pt_write;
}

if (table[index] == shadow_trap_nonpresent_pte) {
@@ -1039,6 +1024,13 @@ static int nonpaging_map(struct kvm_vcpu *vcpu, gva_t v, int write, gfn_t gfn)
page = gfn_to_page(vcpu->kvm, gfn);
up_read(&current->mm->mmap_sem);

+ /* mmio */
+ if (is_error_page(page)) {
+ kvm_release_page_clean(page);
+ up_read(&vcpu->kvm->slots_lock);
+ return 1;

```

[PATCH 11/40] KVM: MMU: Decouple mmio from shadow page tables

```

+ }
+
spin_lock(&vcpu->kvm->mmu_lock);
kvm_mmu_free_some_pages(vcpu);
r = __nonpaging_map(vcpu, v, write, gfn, page);
@@ -1406,10 +1398,14 @@ static void mmu_guess_page_from_pte_write(struct kvm_vcpu *vcpu, gpa_t
gpa,
return;
gfn = (gpte & PT64_BASE_ADDR_MASK) >> PAGE_SHIFT;

- down_read(&current->mm->mmap_sem);
+ down_read(&vcpu->kvm->slots_lock);
page = gfn_to_page(vcpu->kvm, gfn);
- up_read(&current->mm->mmap_sem);
+ up_read(&vcpu->kvm->slots_lock);

+ if (is_error_page(page)) {
+ kvm_release_page_clean(page);
+ return;
+ }
vcpu->arch.update_pte.gfn = gfn;
vcpu->arch.update_pte.page = page;
}
diff --git a/arch/x86/kvm/paging_tmpl.h b/arch/x86/kvm/paging_tmpl.h
index c2fd2b9..4b55f46 100644
--- a/arch/x86/kvm/paging_tmpl.h
+++ b/arch/x86/kvm/paging_tmpl.h
@@ -399,6 +399,14 @@ static int FNAME(page_fault)(struct kvm_vcpu *vcpu, gva_t addr,
page = gfn_to_page(vcpu->kvm, walker.gfn);
up_read(&current->mm->mmap_sem);

+ /* mmio */
+ if (is_error_page(page)) {
+ pgprintk("gfn %x is mmio\n", walker.gfn);
+ kvm_release_page_clean(page);
+ up_read(&vcpu->kvm->slots_lock);
+ return 1;
+ }
+
spin_lock(&vcpu->kvm->mmu_lock);
kvm_mmu_free_some_pages(vcpu);
shadow_pte = FNAME(fetch)(vcpu, addr, &walker, user_fault, write_fault,
@@ -409,15 +417,6 @@ static int FNAME(page_fault)(struct kvm_vcpu *vcpu, gva_t addr,
if (!write_pt)
vcpu->arch.last_pt_write_count = 0; /* reset fork detector */

- /*
- * mmio: emulate if accessible, otherwise its a guest fault.
- */
- if (shadow_pte && is_io_pte(*shadow_pte)) {
- spin_unlock(&vcpu->kvm->mmu_lock);

```

[PATCH 11/40] KVM: MMU: Decouple mmio from shadow page tables

```
- up_read(&vcpu->kvm->slots_lock);  
- return 1;  
- }  
-  
++vcpu->stat.pf_fixed;  
kvm_mmu_audit(vcpu, "post page fault (fixed)");  
spin_unlock(&vcpu->kvm->mmu_lock);  
--
```

1.5.4.5

--
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>