

# [PATCH 15/50] KVM: VMX: Add list of potentially locally cached vcpus

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-06/msg10265.html>

---

- *From:* Avi Kivity <[avi@xxxxxxxxxxxxx](mailto:avi@xxxxxxxxxxxxx)>
  - *Date:* Thu, 26 Jun 2008 15:27:57 +0300
- 

VMX hardware can cache the contents of a vcpu's vmcs. This cache needs to be flushed when migrating a vcpu to another cpu, or (which is the case that interests us here) when disabling hardware virtualization on a cpu.

The current implementation of decaching iterates over the list of all vcpus, picks the ones that are potentially cached on the cpu that is being offlined, and flushes the cache. The problem is that it uses `mutex_trylock()` to gain exclusive access to the vcpu, which fires off a (benign) warning about using the mutex in an interrupt context.

To avoid this, and to make things generally nicer, add a new per-cpu list of potentially cached vcpus. This makes the decaching code much simpler. The list is vmx-specific since other hardware doesn't have this issue.

[andrea: fix crash on suspend/resume]

Signed-off-by: Andrea Arcangeli <[andrea@xxxxxxxxxxxxx](mailto:andrea@xxxxxxxxxxxxx)>

Signed-off-by: Avi Kivity <[avi@xxxxxxxxxxxxx](mailto:avi@xxxxxxxxxxxxx)>

---

arch/x86/kvm/vmx.c | 24 ++++++-----  
arch/x86/kvm/x86.c | 27 -----  
2 files changed, 22 insertions(+), 29 deletions(-)

diff --git a/arch/x86/kvm/vmx.c b/arch/x86/kvm/vmx.c

index fa4ea2e..b99d045 100644

--- a/arch/x86/kvm/vmx.c

+++ b/arch/x86/kvm/vmx.c

@@ -55,6 +55,7 @@ struct vmcs {

struct vcpu\_vmx {

struct kvm\_vcpu vcpu;

+ struct list\_head local\_vcpus\_link;

int launched;

u8 fail;

u32 idt\_vectoring\_info;

@@ -93,6 +94,7 @@ static int init\_rmode(struct kvm \*kvm);

static DEFINE\_PER\_CPU(struct vmcs \*, vmxarea);



[PATCH 15/50] KVM: VMX: Add list of potentially locally cached vcpus

```

+ list_for_each_entry_safe(vmx, n, &per_cpu(vcpus_on_cpu, cpu),
+ local_vcpus_link)
+ __vcpu_clear(vmx);
+}
+
static void hardware_disable(void *garbage)
{
+ vmclear_local_vcpus();
asm volatile (__ex(ASM_VMX_VMXOFF) : : "cc");
write_cr4(read_cr4() & ~X86_CR4_VMXE);
}
@@ -2967,7 +2987,7 @@ static void vmx_free_vmcs(struct kvm_vcpu *vcpu)
struct vcpu_vmx *vmx = to_vmx(vcpu);

if (vmx->vmcs) {
- on_each_cpu(__vcpu_clear, vmx, 0, 1);
+ vcpu_clear(vmx);
free_vmcs(vmx->vmcs);
vmx->vmcs = NULL;
}
diff --git a/arch/x86/kvm/x86.c b/arch/x86/kvm/x86.c
index cc5b0d3..111f432 100644
--- a/arch/x86/kvm/x86.c
+++ b/arch/x86/kvm/x86.c
@@ -823,33 +823,6 @@ out:
*/
void decache_vcpus_on_cpu(int cpu)
{
- struct kvm *vm;
- struct kvm_vcpu *vcpu;
- int i;
-
- spin_lock(&kvm_lock);
- list_for_each_entry(vm, &vm_list, vm_list)
- for (i = 0; i < KVM_MAX_VCPUS; ++i) {
- vcpu = vm->vcpus[i];
- if (!vcpu)
- continue;
- /*
- * If the vcpu is locked, then it is running on some
- * other cpu and therefore it is not cached on the
- * cpu in question.
- *
- * If it's not locked, check the last cpu it executed
- * on.
- */
- if (mutex_trylock(&vcpu->mutex)) {
- if (vcpu->cpu == cpu) {
- kvm_x86_ops->vcpu_decache(vcpu);
- vcpu->cpu = -1;
- }

```

[PATCH 15/50] KVM: VMX: Add list of potentially locally cached vcpus

```
- mutex_unlock(&vcpu->mutex);  
- }  
- }  
- spin_unlock(&kvm_lock);  
}
```

```
int kvm_dev_ioctl_check_extension(long ext)
```

```
--
```

1.5.6

```
--
```

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/>