

[PATCH 33/40] x86: KVM guest: paravirtualized clocksource

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-03/msg12000.html>

- *From:* Avi Kivity <avi@xxxxxxxxxxxxx>
 - *Date:* Mon, 31 Mar 2008 17:37:17 +0300
-

From: Glauber de Oliveira Costa <gcosta@xxxxxxxxxx>

This is the guest part of kvm clock implementation
It does not do tsc-only timing, as tsc can have deltas
between cpus, and it did not seem worthy to me to keep
adjusting them.

We do use it, however, for fine-grained adjustment.

Other than that, time comes from the host.

[randy dunlap: add missing include]

[randy dunlap: disallow on Voyager or Visual WS]

Signed-off-by: Glauber de Oliveira Costa <gcosta@xxxxxxxxxx>

Signed-off-by: Randy Dunlap <randy.dunlap@xxxxxxxxxx>

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

arch/x86/Kconfig | 11 +++
arch/x86/kernel/Makefile | 1 +
arch/x86/kernel/kvmclock.c | 160 +++
arch/x86/kernel/setup_32.c | 5 ++
arch/x86/kernel/setup_64.c | 5 ++
5 files changed, 182 insertions(+), 0 deletions(-)
create mode 100644 arch/x86/kernel/kvmclock.c

diff --git a/arch/x86/Kconfig b/arch/x86/Kconfig

index 6c70fed..e59ea05 100644

--- a/arch/x86/Kconfig

+++ b/arch/x86/Kconfig

@@ -370,6 +370,17 @@ config VMI

at the moment), by linking the kernel to a GPL-ed ROM module
provided by the hypervisor.

+config KVM_CLOCK

+ bool "KVM paravirtualized clock"

+ select PARAVIRT

+ depends on !(X86_VISWS || X86_VOYAGER)

[PATCH 33/40] x86: KVM guest: paravirtualized clocksource

```
+ help
+ Turning on this option will allow you to run a paravirtualized clock
+ when running over the KVM hypervisor. Instead of relying on a PIT
+ (or probably other) emulation by the underlying device model, the host
+ provides the guest with timing infrastructure such as time of day, and
+ system time
+
source "arch/x86/1guest/Kconfig"

config PARAVIRT
diff --git a/arch/x86/kernel/Makefile b/arch/x86/kernel/Makefile
index 4eb5ce8..a3379a3 100644
--- a/arch/x86/kernel/Makefile
+++ b/arch/x86/kernel/Makefile
@@ -77,6 +77,7 @@ obj-$(CONFIG_DEBUG_RODATA_TEST) += test_rodata.o
obj-$(CONFIG_DEBUG_NX_TEST) += test_nx.o

obj-$(CONFIG_VMI) += vmi_32.o vmiclock_32.o
+obj-$(CONFIG_KVM_CLOCK) += kvmclock.o
obj-$(CONFIG_PARAVIRT) += paravirt.o paravirt_patch_$(BITS).o

ifdef CONFIG_INPUT_PCSPKR
diff --git a/arch/x86/kernel/kvmclock.c b/arch/x86/kernel/kvmclock.c
new file mode 100644
index 0000000..b999f5e
--- /dev/null
+++ b/arch/x86/kernel/kvmclock.c
@@ -0,0 +1,160 @@
+/* KVM paravirtual clock driver. A clocksource implementation
+ Copyright (C) 2008 Glauber de Oliveira Costa, Red Hat Inc.
+
+ This program is free software; you can redistribute it and/or modify
+ it under the terms of the GNU General Public License as published by
+ the Free Software Foundation; either version 2 of the License, or
+ (at your option) any later version.
+
+ This program is distributed in the hope that it will be useful,
+ but WITHOUT ANY WARRANTY; without even the implied warranty of
+ MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
+ GNU General Public License for more details.
+
+ You should have received a copy of the GNU General Public License
+ along with this program; if not, write to the Free Software
+ Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
+*/
+
+#include <linux/clocksource.h>
+#include <linux/kvm_para.h>
+#include <asm/arch_hooks.h>
+#include <asm/msr.h>
+#include <asm/apic.h>
```

[PATCH 33/40] x86: KVM guest: paravirtualized clocksource

```
+#include <linux/percpu.h>
+
+#define KVM_SCALE 22
+
+static int kvmclock = 1;
+
+static int parse_no_kvmclock(char *arg)
+{
+ kvmclock = 0;
+ return 0;
+}
+early_param("no-kvmclock", parse_no_kvmclock);
+
+/* The hypervisor will put information about time periodically here */
+static DEFINE_PER_CPU_SHARED_ALIGNED(struct kvm_vcpu_time_info, hv_clock);
+#define get_clock(cpu, field) per_cpu(hv_clock, cpu).field
+
+static inline u64 kvm_get_delta(u64 last_tsc)
+{
+ int cpu = smp_processor_id();
+ u64 delta = native_read_tsc() - last_tsc;
+ return (delta * get_clock(cpu, tsc_to_system_mul)) >> KVM_SCALE;
+}
+
+static struct kvm_wall_clock wall_clock;
+static cycle_t kvm_clock_read(void);
+/*
+ * The wallclock is the time of day when we booted. Since then, some time may
+ * have elapsed since the hypervisor wrote the data. So we try to account for
+ * that with system time
+ */
+unsigned long kvm_get_wallclock(void)
+{
+ u32 wc_sec, wc_nsec;
+ u64 delta;
+ struct timespec ts;
+ int version, nsec;
+ int low, high;
+
+ low = (int)__pa(&wall_clock);
+ high = ((u64)__pa(&wall_clock) >> 32);
+
+ delta = kvm_clock_read();
+
+ native_write_msr(MSR_KVM_WALL_CLOCK, low, high);
+ do {
+ version = wall_clock.wc_version;
+ rmb();
+ wc_sec = wall_clock.wc_sec;
+ wc_nsec = wall_clock.wc_nsec;
+ rmb();
```

[PATCH 33/40] x86: KVM guest: paravirtualized clocksource

```
+ } while ((wall_clock.wc_version != version) || (version & 1));
+
+ delta = kvm_clock_read() - delta;
+ delta += wc_nsec;
+ nsec = do_div(delta, NSEC_PER_SEC);
+ set_normalized_timespec(&ts, wc_sec + delta, nsec);
+ /*
+ * Of all mechanisms of time adjustment I've tested, this one
+ * was the champion!
+ */
+ return ts.tv_sec + 1;
+}
+
+int kvm_set_wallclock(unsigned long now)
+{
+ return 0;
+}
+
+/*
+ * This is our read_clock function. The host puts an tsc timestamp each time
+ * it updates a new time. Without the tsc adjustment, we can have a situation
+ * in which a vcpu starts to run earlier (smaller system_time), but probes
+ * time later (compared to another vcpu), leading to backwards time
+ */
+static cycle_t kvm_clock_read(void)
+{
+ u64 last_tsc, now;
+ int cpu;
+
+ preempt_disable();
+ cpu = smp_processor_id();
+
+ last_tsc = get_clock(cpu, tsc_timestamp);
+ now = get_clock(cpu, system_time);
+
+ now += kvm_get_delta(last_tsc);
+ preempt_enable();
+
+ return now;
+}
+static struct clocksource kvm_clock = {
+ .name = "kvm-clock",
+ .read = kvm_clock_read,
+ .rating = 400,
+ .mask = CLOCKSOURCE_MASK(64),
+ .mult = 1 << KVM_SCALE,
+ .shift = KVM_SCALE,
+ .flags = CLOCK_SOURCE_IS_CONTINUOUS,
+};
+
+static int kvm_register_clock(void)
```

[PATCH 33/40] x86: KVM guest: paravirtualized clocksource

```

+{
+ int cpu = smp_processor_id();
+ int low, high;
+ low = (int)__pa(&per_cpu(hv_clock, cpu)) | 1;
+ high = ((u64)__pa(&per_cpu(hv_clock, cpu)) >> 32);
+
+ return native_write_msr_safe(MSR_KVM_SYSTEM_TIME, low, high);
+}
+
+static void kvm_setup_secondary_clock(void)
+{
+ /*
+ * Now that the first cpu already had this clocksource initialized,
+ * we shouldn't fail.
+ */
+ WARN_ON(kvm_register_clock());
+ /* ok, done with our trickery, call native */
+ setup_secondary_APIC_clock();
+}
+
+void __init kvmclock_init(void)
+{
+ if (!kvm_para_available())
+ return;
+
+ if (kvmclock && kvm_para_has_feature(KVM_FEATURE_CLOCKSOURCE)) {
+ if (kvm_register_clock())
+ return;
+ pv_time_ops.get_wallclock = kvm_get_wallclock;
+ pv_time_ops.set_wallclock = kvm_set_wallclock;
+ pv_time_ops.sched_clock = kvm_clock_read;
+ pv_apic_ops.setup_secondary_clock = kvm_setup_secondary_clock;
+ clocksource_register(&kvm_clock);
+ }
+}
diff --git a/arch/x86/kernel/setup_32.c b/arch/x86/kernel/setup_32.c
index 2b3e5d4..3ef92a2 100644
--- a/arch/x86/kernel/setup_32.c
+++ b/arch/x86/kernel/setup_32.c
@@ -46,6 +46,7 @@
#include <linux/pfn.h>
#include <linux/pci.h>
#include <linux/init_ohci1394_dma.h>
+#include <linux/kvm_para.h>

#include <video/edid.h>

@@ -770,6 +771,10 @@ void __init setup_arch(char **cmdline_p)

max_low_pfn = setup_memory();

```

[PATCH 33/40] x86: KVM guest: paravirtualized clocksource

```
+#ifdef CONFIG_KVM_CLOCK
+ kvmclock_init();
+#endif
+
#ifdef CONFIG_VMI
/*
 * Must be after max_low_pfn is determined, and before kernel
diff --git a/arch/x86/kernel/setup_64.c b/arch/x86/kernel/setup_64.c
index f4f7ecf..26b676f 100644
--- a/arch/x86/kernel/setup_64.c
+++ b/arch/x86/kernel/setup_64.c
@@ -41,6 +41,7 @@
#include <linux/ctype.h>
#include <linux/uaccess.h>
#include <linux/init_ohci1394_dma.h>
#include <linux/kvm_para.h>

#include <asm/mtrr.h>
#include <asm/uaccess.h>
@@ -349,6 +350,10 @@ void __init setup_arch(char **cmdline_p)

io_delay_init();

+#ifdef CONFIG_KVM_CLOCK
+ kvmclock_init();
+#endif
+
#ifdef CONFIG_SMP
/* setup to use the early static init tables during kernel startup */
x86_cpu_to_apicid_early_ptr = (void *)x86_cpu_to_apicid_init;
--
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/>