

nforce2 bugs?

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-09/8916.html>

From: white phoenix (white.phoenix_at_gmail.com)

Date: 09/30/04

Date: Wed, 29 Sep 2004 18:42:23 -0400

To: linux-kernel@vger.kernel.org

I'm wondering if any of the older nforce2 chipset bugs have been fixed in recent kernels. i've seen a number of nforce2 kernel patches floating around. can someone tell me what some of these do? lol.

-----nforce2-disconnect-quirk.patch-----

[x86] fix lockups with APIC support on nForce2

Add PCI quirk to disable Halt Disconnect and Stop Grant Disconnect (based on athcool program by Osamu Kayasono).

Spotted by Prakash K. Cheemplavam <prakashpublic@gmx.de> and Mathieu <cheuche+lkml@free.fr>.

arch/i386/pci/fixup.c | 17 +++++
1 files changed, 17 insertions(+)

```
diff -puN arch/i386/pci/fixup.c~nforce2-disconnect-quirk arch/i386/pci/fixup.c
--- linux-2.6.0-test11/arch/i386/pci/fixup.c~nforce2-disconnect-quirk 2003-12-08
00:09:56.480294672 +0100
+++ linux-2.6.0-test11-root/arch/i386/pci/fixup.c 2003-12-08
00:09:56.484294064 +0100
@@ -187,6 +187,22 @@ static void __devinit pci_fixup_transpar
        dev->transparent = 1;
    }

+/*
+ * Halt Disconnect and Stop Grant Disconnect (bit 4 at offset 0x6F)
+ * must be disabled when APIC is used (or lockups will happen).
+ */
+static void __devinit pci_fixup_nforce2_disconnect(struct pci_dev *d)
+{
+    u8 t;
+
+    pci_read_config_byte(d, 0x6F, &t);
+    if (t & 0x10) {
+        printk(KERN_INFO "PCI: disabling nForce2 Halt Disconnect"
```

Linux-Kernel: nforce2 bugs?

```
+ " and Stop Grant Disconnect\n");
+ pci_write_config_byte(d, 0x6F, (t & 0xef));
+ }
+}
+
struct pci_fixup pcibios_fixups[] = {
    { PCI_FIXUP_HEADER, PCI_VENDOR_ID_INTEL, PCI_DEVICE_ID_INTEL_82451NX,
pci_fixup_i450nx
    },
    { PCI_FIXUP_HEADER, PCI_VENDOR_ID_INTEL, PCI_DEVICE_ID_INTEL_82454GX,
pci_fixup_i450gx
    },
@@ -205,5 +221,6 @@ struct pci_fixup pcibios_fixups[] = {
    { PCI_FIXUP_HEADER, PCI_VENDOR_ID_VIA, PCI_DEVICE_ID_VIA_8367_0,
pci_fixup_via_northbridge_bug
    },
    { PCI_FIXUP_HEADER, PCI_VENDOR_ID_NCR, PCI_DEVICE_ID_NCR_53C810,
pci_fixup_ncr53c810
    },
    { PCI_FIXUP_HEADER, PCI_VENDOR_ID_INTEL, PCI_ANY_ID, pci_fixup_transparent_bridge
    },
+ { PCI_FIXUP_HEADER, PCI_VENDOR_ID_NVIDIA, PCI_DEVICE_ID_NVIDIA_NFORCE2,
pci_fixup_nforce2_disconnect
    },
    { 0 }
};
```

-

-----nforce2-idleC1halt-rd-2.6.5.patch-----

--- linux-2.6.5/arch/i386/kernel/process.c.orig 2004-04-04

13:36:10.000000000 +1000

+++ linux-2.6.5/arch/i386/kernel/process.c 2004-04-15 20:41:13.000000000 +1000

@@ -47,10 +47,13 @@

#include <asm/irq.h>

#include <asm/desc.h>

#ifdef CONFIG_MATH_EMULATION

#include <asm/math_emu.h>

#endif

+#if defined(CONFIG_X86_UP_APIC)

+#include <asm/apic.h>

+#endif

#include <linux/irq.h>

#include <linux/err.h>

asmlinkage void ret_from_fork(void) __asm__("ret_from_fork");

@@ -98,10 +101,34 @@ void default_idle(void)

local_irq_enable();

}

nforce2 bugs?

Linux–Kernel: nforce2 bugs?

```
}

/*
+ * We use this to avoid nforce2 lockups
+ * Reduces frequency of C1 disconnects
+ */
+static void c1halt_idle(void)
+{
+ if (!hlt_counter && current_cpu_data.hlt_works_ok) {
+ local_irq_disable();
+ #if defined(CONFIG_X86_UP_APIC)
+ /* only hlt disconnect if more than 1.6% of apic interval remains */
+ extern int enable_local_apic;
+ if(!need_resched() && (enable_local_apic < 0 ||
+ (apic_read(APIC_TMCCT) > (apic_read(APIC_TMICT)>>6)))) {
+ #else
+ /* just adds a little delay to assist in back to back disconnects */
+ if(!need_resched()) {
+ #endif
+ ndelay(600); /* helps nforce2 but adds 0.6us hard int latency */
+ safe_halt(); /* nothing better to do until we wake up */
+ } else {
+ local_irq_enable();
+ }
+ }
+ }
+ /*
+ * On SMP it's slightly faster (but much more power–consuming!)
+ * to poll the →work.need_resched flag instead of waiting for the
+ * cross–CPU IPI to arrive. Use this option with caution.
+ */
static void poll_idle (void)
@@ –135,20 +162,18 @@ static void poll_idle (void)
+ * The idle thread. There's no useful work to be
+ * done, so just try to conserve power and have a
+ * low exit latency (ie sit in a loop waiting for
+ * somebody to say that they'd like to reschedule)
+ */
+static void (*idle)(void);
void cpu_idle (void)
{
    /* endless idle loop with no priority at all */
    while (1) {
        while (!need_resched()) {
– void (*idle)(void) = pm_idle;
–
–         if (!idle)
– idle = default_idle;
–
+ idle = pm_idle ? pm_idle : default_idle;
        irq_stat[smp_processor_id()].idle_timestamp = jiffies;
```

Linux-Kernel: nforce2 bugs?

```
        idle();
    }
    schedule();
}
@@ -199,16 +224,18 @@ void __init select_idle_routine(const st

static int __init idle_setup (char *str)
{
    if (!strncmp(str, "poll", 4)) {
        printk("using polling idle threads.\n");
- pm_idle = poll_idle;
+ idle = poll_idle;
    } else if (!strncmp(str, "halt", 4)) {
        printk("using halt in idle threads.\n");
- pm_idle = default_idle;
+ idle = default_idle;
+ } else if (!strncmp(str, "C1halt", 6)) {
+ printk("using C1 halt disconnect friendly idle threads.\n");
+ idle = c1halt_idle;
    }
-
    return 1;
}

__setup("idle=", idle_setup);
```

-----nforce2-ioapic-rd-2.6.5.patch-----

```
--- linux-2.6.5/arch/i386/kernel/io_apic.c.orig 2004-04-16
00:20:54.000000000 +1000
+++ linux-2.6.5/arch/i386/kernel/io_apic.c 2004-04-15 20:24:18.000000000 +1000
@@ -2179,10 +2179,13 @@ static inline void check_timer(void)
```

```
    if (pin1 != -1) {
        /*
         * Ok, does IRQ0 through the IOAPIC work?
         */
+ extern int acpi_skip_timer_override;
+ if(acpi_skip_timer_override)
+ timer_ack=0;
        unmask_IO_APIC_irq(0);
        if (timer_irq_works()) {
            if (nmi_watchdog == NMI_IO_APIC) {
                disable_8259A_irq(0);
                setup_nmi();
```

-----nforce-apic-tack.patch-----

```
--- linux-2.6.0/arch/i386/kernel/apic.c 2003-12-18 12:59:58.000000000 +1000
+++ linux-2.6.0-rd/arch/i386/kernel/apic.c 2003-12-21 12:39:28.000000000 +1000
@@ -1070,10 +1070,17 @@ inline void smp_local_timer_interrupt(st
```

Linux–Kernel: nforce2 bugs?

```
    * we can take more than 100K local irqs per second on a 100 MHz P5.
    */
}

/*
+ * Athlon nforce2 R.D.
+ * preset timer ack mode if desired
+ * e.g. static int apic_timerack = 2;
+*/
+static int apic_timerack;
+
+/*
+ * Local APIC timer interrupt. This is the most natural way for doing
+ * local interrupts, but local timer interrupts can be emulated by
+ * broadcast interrupts too. [in case the hw doesn't support APIC timers]
+ *
+ * [ if a single–CPU system runs an SMP kernel then we call the local
+ * @@ –1088,10 +1095,54 @@ void smp_apic_timer_interrupt(struct pt_
+ * the NMI deadlock–detector uses this.
+ */
+   irq_stat[cpu].apic_timer_irqs++;

/*
+ * Athlon nforce2 timer ack delay. Ross Dickson.
+ * works around issue of hard lockups in code location
+ * where linux exposes underlying system timing fault?
+ * hopefully manufacturers will fix it soon.
+ * We leave C1 disconnect bit alone as bios/SMM wants?
+ */
+ if(apic_timerack) {
+ if(apic_timerack==1) {
+ /* v1 timer ack delay, inline delay version
+ * on AMDXP & nforce2 chipset we use at least 500ns
+ * try to scale delay time with cpu speed.
+ * safe all cpu cores?
+ */
+ ndelay((cpu_khz >> 12)+200); /* don't ack too soon or hard lockup */
+ } else {
+ static unsigned int passno, safecnt;
+ /* v2 timer ack delay, timeout version, more efficient
+ * on AMDXP & nforce2 chipset we need 800ns?
+ * from timer irq start to apic irq ack, read apic timer,
+ * may be unsafe for thoroughbred cores?
+ */
+ if(!passno) { /* calculate timing */
+ safecnt = apic_read(APIC_TMICT) –
+ ( (800UL * apic_read(APIC_TMICT) ) /
+ (1000000000UL/HZ) );
+ printk("..APIC TIMER ack delay, reload:%lu, safe:%u\n",
+ apic_read(APIC_TMICT), safecnt);
+ passno++;

```

nforce2 bugs?

Linux–Kernel: nforce2 bugs?

```
+ }
+#if APIC_DEBUG
+ if(passno<12) {
+ unsigned int at1 = apic_read(APIC_TMCCT);
+ if( passno > 1 )
+ Dprintf("..APIC TIMER ack delay, predelay count:%u \n", at1 );
+ passno++;
+ }
+# endif
+ /* delay only if required */
+ while( apic_read(APIC_TMCCT) > safecnt )
+ ndelay(100);
+ }
+ }
+
+ /*
+  * NOTE! We'd better ACK the irq immediately,
+  * because timer handling can be slow.
+  */
+ ack_APIC_irq();
+ /*
@@ -1157,10 +1208,28 @@ asmlinkage void smp_error_interrupt(void
+ smp_processor_id(), v , v1);
+ irq_exit();
+ }

+ /*
+ * Athlon nforce2 timer ack delay. R.D.
+ * kernel arg apic_tack=[012]
+ * 0 off, 1 always delay, 2 timeout
+ */
+static int __init setup_apic_timerack(char *str)
+{
+ int tack;
+
+ get_option(&str, &tack);
+
+ if ( tack < 0 || tack > 2 )
+ return 0;
+ apic_timerack = tack;
+ return 1;
+}
+__setup("apic_tack=", setup_apic_timerack);
+
+ /*
+  * This initializes the IO–APIC and APIC hardware if this is
+  * a UP kernel.
+  */
+int __init APIC_init_uniprocessor (void)
+{
+ --- CUT HERE ---
```

Linux–Kernel: nforce2 bugs?

io–apic edge:

---- CUT HERE ----

--- linux–2.6.0/arch/i386/kernel/io_apic.c 2003–12–18 12:58:39.000000000 +1000

+++ linux–2.6.0–rd/arch/i386/kernel/io_apic.c 2003–12–20

21:41:52.000000000 +1000

@@ –2123,12 +2123,56 @@ static inline void check_timer(void)

 check_nmi_watchdog();

 }

 return;

 }

 clear_IO_APIC_pin(0, pin1);

– printk(KERN_ERR "..MP–BIOS bug: 8254 timer not connected to IO–APIC\n");

+ printk(KERN_ERR "..MP–BIOS bug: 8254 timer not connected to IO–APIC

INTIN%d\n", pin1);

+ }

+

+ #if defined(CONFIG_ACPI_BOOT) && defined(CONFIG_X86_UP_IOAPIC)

+ /* for nforce2 try vector 0 on pin0

+ * Note 8259a is already masked, also by default

+ * the io_apic_set_pci_routing call disables the 8259 irq 0

+ * so we must be connected directly to the 8254 timer if this works

+ * Note2: this violates the above comment re Subtle but works!

+ */

+ printk(KERN_INFO "..TIMER: Is timer irq0 connected to IO–APIC INTIN0? …\n");

+ if (pin1 != –1) {

+ extern spinlock_t i8259A_lock;

+ unsigned long flags;

+ int tok, saved_timer_ack = timer_ack;

+ /*

+ * Ok, does IRQ0 through the IOAPIC work?

+ */

+ io_apic_set_pci_routing (0, 0, 0, 0, 0); /* connect pin */

+ unmask_IO_APIC_irq(0);

+ timer_ack = 0;

+

+ /*

+ * Ok, does IRQ0 through the IOAPIC work?

+ */

+ spin_lock_irqsave(&i8259A_lock, flags);

+ Dprintf("..TIMER 8259A ints disabled?, imr1:%02x, imr2:%02x\n",

inb(0x21), inb(0xA1));

+ tok = timer_irq_works();

+ spin_unlock_irqrestore(&i8259A_lock, flags);

+ if (tok) {

+ if (nmi_watchdog == NMI_IO_APIC) {

+ disable_8259A_irq(0);

+ setup_nmi();

+ enable_8259A_irq(0);

+ check_nmi_watchdog();

+ }

+ printk(KERN_INFO "..TIMER: works OK on IO–APIC INTIN0 irq0\n");

nforce2 bugs?

Linux-Kernel: nforce2 bugs?

```
+ return;
+ }
+ /* failed */
+ timer_ack = saved_timer_ack;
+ clear_IO_APIC_pin(0, 0);
+ io_apic_set_pci_routing ( 0, pin1, 0, 0, 0);
+ printk(KERN_ERR "..MP-BIOS: 8254 timer not connected to IO-APIC INTIN0\n");
+ }
+#endif
```

```
printk(KERN_INFO "...trying to set up timer (IRQ0) through the 8259A ... ");
if (pin2 != -1) {
    printk("\n..... (found pin %d) ...", pin2);
    /*
```

-----more-nforce-fixes.patch-----

```
===== Documentation/kernel-parameters.txt 1.44 vs edited =====
--- 1.44/Documentation/kernel-parameters.txt Mon Mar 22 16:03:22 2004
+++ edited/Documentation/kernel-parameters.txt Wed Apr 21 15:28:12 2004
@@ -122,6 +122,10 @@
```

acpi_serialize [HW,ACPI] force serialization of AML methods

```
+ acpi_skip_timer_override [HW,ACPI]
+ Recognize and ignore IRQ0/pin2 Interrupt Override.
+ For broken nForce2 BIOS resulting in XT-PIC timer.
+
```

ad1816= [HW,OSS]

Format: <io>,<irq>,<dma>,<dma2>

See also Documentation/sound/oss/AD1816.

```
===== arch/i386/kernel/dmi_scan.c 1.57 vs edited =====
--- 1.57/arch/i386/kernel/dmi_scan.c Fri Apr 16 22:03:06 2004
+++ edited/arch/i386/kernel/dmi_scan.c Wed Apr 21 18:29:35 2004
@@ -540,6 +540,19 @@
#endif
```

```
/*
+ * early nForce2 reference BIOS shipped with a
+ * bogus ACPI IRQ0 -> pin2 interrupt override -- ignore it
+ */
+static __init int ignore_timer_override(struct dmi_blacklist *d)
+{
+ extern int acpi_skip_timer_override;
+ printk(KERN_NOTICE "%s detected: BIOS IRQ0 pin2 override"
+ " will be ignored\n", d->ident);
+
+ acpi_skip_timer_override = 1;
+ return 0;
+}
+/*
```

nforce2 bugs?

Linux–Kernel: nforce2 bugs?

```
* Process the DMI blacklists
```

```
*/
```

```
@@ -944,6 +957,37 @@
```

```
    MATCH(DMI_BOARD_VENDOR, "IBM"),  
    MATCH(DMI_PRODUCT_NAME, "eserver xSeries 440"),  
    NO_MATCH, NO_MATCH }},
```

```
+
```

```
+/*
```

```
+ * Systems with nForce2 BIOS timer override bug
```

```
+ * add Albatron KM18G Pro
```

```
+ * add DFI NFII 400–AL
```

```
+ * add Epox 8RGA+
```

```
+ * add Shuttle AN35N
```

```
+ */
```

```
+ { ignore_timer_override, "Abit NF7–S v2", {  
+ MATCH(DMI_BOARD_VENDOR, "http://www.abit.com.tw/"),  
+ MATCH(DMI_BOARD_NAME, "NF7–S/NF7,NF7–V (nVidia–nForce2)" ),  
+ MATCH(DMI_BIOS_VERSION, "6.00 PG"),  
+ MATCH(DMI_BIOS_DATE, "03/24/2004") }},  
+  
+ { ignore_timer_override, "Asus A7N8X v2", {  
+ MATCH(DMI_BOARD_VENDOR, "ASUSTeK Computer INC."),  
+ MATCH(DMI_BOARD_NAME, "A7N8X2.0"),  
+ MATCH(DMI_BIOS_VERSION, "ASUS A7N8X2.0 Deluxe ACPI BIOS Rev 1007"),  
+ MATCH(DMI_BIOS_DATE, "10/06/2003") }},  
+  
+ { ignore_timer_override, "Asus A7N8X–X", {  
+ MATCH(DMI_BOARD_VENDOR, "ASUSTeK Computer INC."),  
+ MATCH(DMI_BOARD_NAME, "A7N8X–X"),  
+ MATCH(DMI_BIOS_VERSION, "ASUS A7N8X–X ACPI BIOS Rev 1007"),  
+ MATCH(DMI_BIOS_DATE, "10/07/2003") }},  
+  
+ { ignore_timer_override, "Shuttle SN41G2", {  
+ MATCH(DMI_BOARD_VENDOR, "Shuttle Inc"),  
+ MATCH(DMI_BOARD_NAME, "FN41"),  
+ MATCH(DMI_BIOS_VERSION, "6.00 PG"),  
+ MATCH(DMI_BIOS_DATE, "01/14/2004") }},  
#endif // CONFIG_ACPI_BOOT
```

```
#ifdef CONFIG_ACPI_PCI
```

```
===== arch/i386/kernel/setup.c 1.115 vs edited =====
```

```
--- 1.115/arch/i386/kernel/setup.c Fri Apr 2 07:21:43 2004
```

```
+++ edited/arch/i386/kernel/setup.c Wed Apr 21 15:28:12 2004
```

```
@@ -614,6 +614,9 @@
```

```
    else if (!memcmp(from, "acpi_sci=low", 12))  
        acpi_sci_flags.polarity = 3;
```

```
+ else if (!memcmp(from, "acpi_skip_timer_override", 24))
```

```
+ acpi_skip_timer_override = 1;
```

```
+
```

nforce2 bugs?

Linux-Kernel: nforce2 bugs?

```
#ifdef CONFIG_X86_LOCAL_APIC
/* disable IO-APIC */
else if (!memcmp(from, "noapic", 6))
===== arch/i386/kernel/acpi/boot.c 1.58 vs edited =====
--- 1.58/arch/i386/kernel/acpi/boot.c Tue Apr 20 20:54:03 2004
+++ edited/arch/i386/kernel/acpi/boot.c Wed Apr 21 15:28:13 2004
@@ -62,6 +62,7 @@

acpi_interrupt_flags acpi_sci_flags __initdata;
int acpi_sci_override_gsi __initdata;
+int acpi_skip_timer_override __initdata;

#ifdef CONFIG_X86_LOCAL_APIC
static u64 acpi_lapic_addr __initdata = APIC_DEFAULT_PHYS_BASE;
@@ -327,6 +328,12 @@
    acpi_sci_ioapic_setup(intsrc->global_irq,
        intsrc->flags.polarity, intsrc->flags.trigger);
    return 0;
+ }
+
+ if (acpi_skip_timer_override &&
+ intsrc->bus_irq == 0 && intsrc->global_irq == 2) {
+ printk(PREFIX "BIOS IRQ0 pin2 override ignored.\n");
+ return 0;
    }

    mp_override_legacy_irq (
===== include/asm-i386/acpi.h 1.18 vs edited =====
--- 1.18/include/asm-i386/acpi.h Tue Mar 30 17:05:19 2004
+++ edited/include/asm-i386/acpi.h Wed Apr 21 15:28:14 2004
@@ -118,6 +118,7 @@
#ifdef CONFIG_X86_IO_APIC
extern int skip_ioapic_setup;
extern int acpi_irq_to_vector(u32 irq); /* deprecated in favor of
acpi_gsi_to_irq */
+extern int acpi_skip_timer_override;

static inline void disable_ioapic_setup(void)
{
-
```

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