

Re: patch x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affi added to -mm tree

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-04/7490.html>

From: Ashok Raj (ashok.raj_at_intel.com)

Date: 04/30/05

Date: Fri, 29 Apr 2005 16:41:22 -0700

To: akpm@osdl.org

Hi Andrew

Sorry for the trouble. In the patch earlier, i missed adding a file that would have broke ia64 builds. Also this time i moved one more common function into generic hardirq framework, and deleted that from i386 and ia64 files.

Updated patch is attached for -mm.

On Fri, Apr 29, 2005 at 05:33:02PM +0200, Andi Kleen wrote:

- >
- > *This looks all very complicated and more than a chipset bug*
- > *than a feature. But anyways.. Cant you just cause a dummy interrupt*
- > *during the reprogramming and not handle it until you reprogrammed?*

Iam not aware how to generate a fake interrupt, and know that this (fake intr) is the interrupt firing, so we can do programming deferred. I will keep looking if there are better ways to do it.

- >
- > *I think that would be much preferable than to add a lot of*
- > *code to the interrupt fast path just for this workaround.*
- >
- > *-Andi*

--
Cheers,
Ashok Raj
- Open Source Technology Center

Signed-off-by: Ashok Raj <ashok.raj@intel.com>

When handling writes to /proc/irq, current code is re-programming rte entries directly. This is not recommended and could potentially cause chipset's to lockup, or cause missing interrupts.

CONFIG_IRQ_BALANCE does this correctly, where it re-programs only when the interrupt is pending. The same needs to be done for /proc/irq handling as well.

Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to 4mm tree

Linux-Kernel: Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to -mm

Otherwise user space irq balancers are really not doing the right thing.

- Changed pending_irq_balance_cpumask to pending_irq_migrate_cpumask for lack of a generic name.
- added move_irq out of IRQ_BALANCE, and added this same to X86_64
- Added new proc handler for write, so we can do deferred write at irq handling time.
- Display of /proc/irq/XX/smp_affinity used to display CPU_MASKALL, instead it now shows only active cpu masks, or exactly what was set.
- Provided a common move_irq implementation, instead of duplicating when using generic irq framework.

Tested on i386/x86_64 and ia64 with CONFIG_PCI_MSI turned on and off.

Tested UP builds as well.

MSI testing: tbd: i have cards, need to look for a x-over cable, although i did test an earlier version of this patch. Will test in a couple days.

```
linux-2.6.12-rc2-mm3-araj/arch/i386/Kconfig | 5
linux-2.6.12-rc2-mm3-araj/arch/i386/kernel/io_apic.c | 37 +++---
linux-2.6.12-rc2-mm3-araj/arch/ia64/Kconfig | 5
linux-2.6.12-rc2-mm3-araj/arch/ia64/kernel/irq.c | 39 -----
linux-2.6.12-rc2-mm3-araj/arch/x86_64/Kconfig | 5
linux-2.6.12-rc2-mm3-araj/arch/x86_64/kernel/io_apic.c | 97 ++++++++-----
linux-2.6.12-rc2-mm3-araj/drivers/pci/msi.c | 17 --
linux-2.6.12-rc2-mm3-araj/drivers/pci/msi.h | 5
linux-2.6.12-rc2-mm3-araj/include/asm-ia64/hw_irq.h | 7 -
linux-2.6.12-rc2-mm3-araj/include/asm-ia64/irq.h | 6 -
linux-2.6.12-rc2-mm3-araj/include/linux/irq.h | 86 ++++++++
linux-2.6.12-rc2-mm3-araj/kernel/irq/manage.c | 8 +
linux-2.6.12-rc2-mm3-araj/kernel/irq/proc.c | 19 +-
13 files changed, 212 insertions(+), 124 deletions(-)
```

```
diff -puN arch/i386/kernel/io_apic.c~fix_irq_affinity arch/i386/kernel/io_apic.c
```

```
--- linux-2.6.12-rc2-mm3/arch/i386/kernel/io_apic.c~fix_irq_affinity 2005-04-28 23:51:28.000000
```

```
+++ linux-2.6.12-rc2-mm3-araj/arch/i386/kernel/io_apic.c 2005-04-29 15:07:33.000000000 -07
```

```
@@ -31,8 +31,8 @@
```

```
#include <linux/mc146818rtc.h>
#include <linux/compiler.h>
#include <linux/acpi.h>
```

```
-
```

```
#include <linux/sysdev.h>
+#include <linux/irq.h>
#include <asm/io.h>
#include <asm/smp.h>
#include <asm/desc.h>
@@ -227,7 +227,14 @@ static void set_ioapic_affinity_irq(unsig
```

```
int pin;
struct irq_pin_list *entry = irq_2_pin + irq;
unsigned int apicid_value;
+ cpumask_t tmp;
```

```
+ cpus_and(tmp, cpumask, cpu_online_map);
+ if (cpus_empty(tmp))
+     tmp = TARGET_CPUS;
+ cpus_and(cpumask, tmp, CPU_MASK_ALL);
```

```
+ apicid_value = cpu_mask_to_apicid(cpumask);
+ /* Prepare to do the io_apic_write */
+ apicid_value = apicid_value << 24;
```

```
@@ -241,9 +248,12 @@ static void set_ioapic_affinity_irq(unsig
break;
entry = irq_2_pin + entry->next;
```

```
}
+ set_irq_info(irq, cpumask);
```

Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to 2mm tree


```

#ifdef CONFIG_SMP
void fastcall send_IPI_self(int vector)
@@ -1247,6 +1245,7 @@ static void __init setup_IO_APIC_irqs(vo
    spin_lock_irqsave(&ioapic_lock, flags);
    io_apic_write(apic, 0x11+2*pin, *(((int *)&entry)+1));
    io_apic_write(apic, 0x10+2*pin, *(((int *)&entry)+0));
+
    set_native_irq_info(irq, TARGET_CPUS);
    spin_unlock_irqrestore(&ioapic_lock, flags);
}
}
@@ -1941,6 +1940,7 @@ static void ack_edge_ioapic_vector(unsigned
{
    int irq = vector_to_irq(vector);

+
    move_irq(vector);
    ack_edge_ioapic_irq(irq);
}

@@ -1955,6 +1955,7 @@ static void end_level_ioapic_vector (unsig
{
    int irq = vector_to_irq(vector);

+
    move_irq(vector);
    end_level_ioapic_irq(irq);
}

@@ -1977,6 +1978,7 @@ static void set_ioapic_affinity_vector (
{
    int irq = vector_to_irq(vector);

+
    set_native_irq_info(vector, cpu_mask);
    set_ioapic_affinity_irq(irq, cpu_mask);
}
#endif
@@ -2566,6 +2568,7 @@ int io_apic_set_pci_routing (int ioapic,
    spin_lock_irqsave(&ioapic_lock, flags);
    io_apic_write(ioapic, 0x11+2*pin, *(((int *)&entry)+1));
    io_apic_write(ioapic, 0x10+2*pin, *(((int *)&entry)+0));
+
    set_native_irq_info(use_pci_vector() ? entry.vector : irq, TARGET_CPUS);
    spin_unlock_irqrestore(&ioapic_lock, flags);

    return 0;
diff -puN drivers/pci/msi.c~fix_irq_affinity drivers/pci/msi.c
--- linux-2.6.12-rc2-mm3/drivers/pci/msi.c~fix_irq_affinity 2005-04-28 23:51:28.000000000 -0700
+++ linux-2.6.12-rc2-mm3-araj/drivers/pci/msi.c 2005-04-29 14:05:23.000000000 -0700
@@ -91,6 +91,7 @@ static void set_msi_affinity(unsigned in
{
    struct msi_desc *entry;
    struct msg_address address;
+
    unsigned int irq = vector;

    entry = (struct msi_desc *)msi_desc[vector];
    if (!entry || !entry->dev)
@@ -112,6 +113,7 @@ static void set_msi_affinity(unsigned in
    entry->msi_attrib.current_cpu = cpu_mask_to_apicid(cpu_mask);
    pci_write_config_dword(entry->dev, msi_lower_address_reg(pos),
        address.lo_address.value);
+
    set_native_irq_info(irq, cpu_mask);
    break;
}
case PCI_CAP_ID_MSIX:

```


Linux-Kernel: Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to -mm

```
* MSI-X Address Register
*/
diff -puN include/linux/irq.h~fix_irq_affinity include/linux/irq.h
--- linux-2.6.12-rc2-mm3/include/linux/irq.h~fix_irq_affinity 2005-04-28 23:51:28.000000000 -07
+++ linux-2.6.12-rc2-mm3-araj/include/linux/irq.h 2005-04-29 15:43:18.000000000 -0700
@@ -71,12 +71,98 @@ typedef struct irq_desc {

extern irq_desc_t irq_desc [NR_IRQS];

+/* Return a pointer to the irq descriptor for IRQ. */
+static inline irq_desc_t *
+irq_descp (int irq)
+{
+    return irq_desc + irq;
+}
+
+
#include <asm/hw_irq.h> /* the arch dependent stuff */

extern int setup_irq(unsigned int irq, struct irqaction * new);

#ifdef CONFIG_GENERIC_HARDIRQS
extern cpumask_t irq_affinity[NR_IRQS];
+
+#ifdef CONFIG_SMP
+static inline void set_native_irq_info(int irq, cpumask_t mask)
+{
+    irq_affinity[irq] = mask;
+}
+#else
+static inline void set_native_irq_info(int irq, cpumask_t mask)
+{
+}
+#endif
+
+#ifdef CONFIG_GENERIC_PENDING_IRQ
+extern cpumask_t pending_irq_cpumask[NR_IRQS];
+
+static inline void
+move_native_irq(int irq)
+{
+    cpumask_t tmp;
+    irq_desc_t *desc = irq_descp(irq);
+
+    if (likely(cpus_empty(pending_irq_cpumask[irq])))
+        return;
+
+    if (unlikely(!desc->handler->set_affinity))
+        return;
+
+    /* note - we hold the desc->lock */
+    cpus_and(tmp, pending_irq_cpumask[irq], cpu_online_map);
+
+    /*
+     * If there was a valid mask to work with, please
+     * do the disable, re-program, enable sequence.
+     * This is *not* particularly important for level triggered
+     * but in a edge trigger case, we might be setting rte
+     * when an active trigger is coming in. This could
+     * cause some ioapics to mal-function.
+     * Being paranoid i guess!
+     */
+}

```

Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to 6mm tree

Linux-Kernel: Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to -mm

```
+     if (unlikely(!cpus_empty(tmp))) {
+         desc->handler->disable(irq);
+         desc->handler->set_affinity(irq,tmp);
+         desc->handler->enable(irq);
+     }
+     cpus_clear(pending_irq_cpumask[irq]);
+}
+
+#ifdef CONFIG_PCI_MSI
+/*
+ * Wonder why these are dummies?
+ * For e.g the set_ioapic_affinity_vector() calls the set_ioapic_affinity_irq()
+ * counter part after translating the vector to irq info. We need to perform
+ * this operation on the real irq, when we dont use vector, i.e when
+ * pci_use_vector() is false.
+ */
+static inline void move_irq(int irq)
+{
+}
+
+static inline void set_irq_info(int irq, cpumask_t mask)
+{
+}
+#else
+static inline void move_irq(int irq)
+{
+     move_native_irq(irq);
+}
+
+static inline void set_irq_info(int irq, cpumask_t mask)
+{
+     set_native_irq_info(irq, mask);
+}
+#endif // CONFIG_PCI_MSI
+#else
+#define move_irq(x)
+#define move_native_irq(x)
+#endif // CONFIG_GENERIC_PENDING_IRQ
+
+extern int no_irq_affinity;
+extern int noirqdebug_setup(char *str);
```

```
diff -puN kernel/irq/manage.c~fix_irq_affinity kernel/irq/manage.c
--- linux-2.6.12-rc2-mm3/kernel/irq/manage.c~fix_irq_affinity    2005-04-28 23:51:28.000000000 -07
+++ linux-2.6.12-rc2-mm3-araaj/kernel/irq/manage.c              2005-04-29 07:00:09.000000000 -0700
@@ -17,6 +17,10 @@
```

```
cpumask_t irq_affinity[NR_IRQS] = { [0 ... NR_IRQS-1] = CPU_MASK_ALL };
```

```
#ifdef CONFIG_GENERIC_PENDING_IRQ
+cpumask_t __cacheline_aligned pending_irq_cpumask[NR_IRQS];
+#endif
```

```
+
+/**
+ * synchronize_irq - wait for pending IRQ handlers (on other CPUs)
+ *
```

```
@@ -36,6 +40,10 @@ void synchronize_irq(unsigned int irq)
```

```
EXPORT_SYMBOL(synchronize_irq);
```

```
#else
+void set_irq_info(unsigned int irq, cpumask_t mask)
```

Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to 7mm tree

Linux-Kernel: Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to -mm

```
+{
+}
#endif

/**
diff -puN arch/ia64/kernel/irq.c~fix_irq_affinity arch/ia64/kernel/irq.c
--- linux-2.6.12-rc2-mm3/arch/ia64/kernel/irq.c~fix_irq_affinity      2005-04-28 23:51:28.00000
+++ linux-2.6.12-rc2-mm3-araj/arch/ia64/kernel/irq.c      2005-04-29 07:44:00.000000000 -0700
@@ -91,23 +91,8 @@ skip:
 }

#ifdef CONFIG_SMP
-/*
- * This is updated when the user sets irq affinity via /proc
- */
-static cpumask_t __cacheline_aligned pending_irq_cpumask[NR_IRQS];
-static unsigned long pending_irq_redir[BITS_TO_LONGS(NR_IRQS)];
-
static char irq_redir [NR_IRQS]; // = { [0 ... NR_IRQS-1] = 1 };

-/*
- * Arch specific routine for deferred write to iosapic rte to reprogram
- * intr destination.
- */
-void proc_set_irq_affinity(unsigned int irq, cpumask_t mask_val)
-{
-    pending_irq_cpumask[irq] = mask_val;
-}
-
void set_irq_affinity_info (unsigned int irq, int hwid, int redir)
{
    cpumask_t mask = CPU_MASK_NONE;
@@ -116,32 +101,10 @@ void set_irq_affinity_info (unsigned int

    if (irq < NR_IRQS) {
        irq_affinity[irq] = mask;
+        set_irq_info(irq, mask);
        irq_redir[irq] = (char) (redir & 0xff);
    }
}

-void move_irq(int irq)
-{
-    /* note - we hold desc->lock */
-    cpumask_t tmp;
-    irq_desc_t *desc = irq_descp(irq);
-    int redir = test_bit(irq, pending_irq_redir);
-
-    if (unlikely(!desc->handler->set_affinity))
-        return;
-
-    if (!cpus_empty(pending_irq_cpumask[irq])) {
-        cpus_and(tmp, pending_irq_cpumask[irq], cpu_online_map);
-        if (unlikely(!cpus_empty(tmp))) {
-            desc->handler->set_affinity(irq | (redir ? IA64_IRQ_REDIRECTED : 0),
-                                     pending_irq_cpumask[irq]);
-        }
-        cpus_clear(pending_irq_cpumask[irq]);
-    }
-}
-
```

Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to 8mm tree

Linux-Kernel: Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to -mm

```
-
#endif /* CONFIG_SMP */

#ifdef CONFIG_HOTPLUG_CPU
diff -puN arch/x86_64/kernel/io_apic.c~fix_irq_affinity arch/x86_64/kernel/io_apic.c
--- linux-2.6.12-rc2-mm3/arch/x86_64/kernel/io_apic.c~fix_irq_affinity 2005-04-28 23:51:28.000000
+++ linux-2.6.12-rc2-mm3-araaj/arch/x86_64/kernel/io_apic.c 2005-04-29 15:07:13.000000000 -07
@@ -75,6 +75,59 @@ int vector_irq[NR_VECTORS] = { [0 ... NR
#define vector_to_irq(vector) (vector)
#endif

+#define __DO_ACTION(R, ACTION, FINAL) \
+ \
+{ \
+    int pin; \
+    struct irq_pin_list *entry = irq_2_pin + irq; \
+ \
+    for (;;) { \
+        unsigned int reg; \
+        pin = entry->pin; \
+        if (pin == -1) \
+            break; \
+        reg = io_apic_read(entry->apic, 0x10 + R + pin*2); \
+        reg ACTION; \
+        io_apic_modify(entry->apic, reg); \
+        if (!entry->next) \
+            break; \
+        entry = irq_2_pin + entry->next; \
+    } \
+    FINAL; \
+}
+
+#ifdef CONFIG_SMP
+static void set_ioapic_affinity_irq(unsigned int irq, cpumask_t mask)
+{
+    unsigned long flags;
+    unsigned int dest;
+    cpumask_t tmp;
+
+    cpus_and(tmp, mask, cpu_online_map);
+    if (cpus_empty(tmp))
+        tmp = TARGET_CPUS;
+
+    cpus_and(mask, tmp, CPU_MASK_ALL);
+
+    dest = cpu_mask_to_apicid(mask);
+
+    /*
+     * Only the high 8 bits are valid.
+     */
+    dest = SET_APIC_LOGICAL_ID(dest);
+
+    spin_lock_irqsave(&ioapic_lock, flags);
+    __DO_ACTION(1, = dest, )
+    set_irq_info(irq, mask);
+    spin_unlock_irqrestore(&ioapic_lock, flags);
+}
+#else
+static void set_ioapic_affinity_irq(unsigned int irq, cpumask_t mask)
+{
+    return;
+}

```

Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to 9mm tree

```

+#endif
+
+ /*
+  * The common case is 1:1 IRQ<->pin mappings. Sometimes there are
+  * shared ISA-space IRQs, so we have to support them. We are super
@@ -98,26 +151,6 @@ static void add_pin_to_irq(unsigned int
    entry->pin = pin;
}

-#define __DO_ACTION(R, ACTION, FINAL)
-
- {
-     int pin;
-     struct irq_pin_list *entry = irq_2_pin + irq;
-
-     for (;;) {
-         unsigned int reg;
-         pin = entry->pin;
-         if (pin == -1)
-             break;
-         reg = io_apic_read(entry->apic, 0x10 + R + pin*2);
-         reg ACTION;
-         io_apic_modify(entry->apic, reg);
-         if (!entry->next)
-             break;
-         entry = irq_2_pin + entry->next;
-     }
-     FINAL;
- }

#define DO_ACTION(name,R,ACTION, FINAL)

@@ -764,6 +797,7 @@ static void __init setup_IO_APIC_irqs(vo
    spin_lock_irqsave(&ioapic_lock, flags);
    io_apic_write(apic, 0x11+2*pin, *(((int *)&entry)+1));
    io_apic_write(apic, 0x10+2*pin, *(((int *)&entry)+0));
+
+     set_native_irq_info(irq, TARGET_CPUS);
    spin_unlock_irqrestore(&ioapic_lock, flags);
}
}

@@ -1312,6 +1346,7 @@ static unsigned int startup_edge_ioapic_
*/
static void ack_edge_ioapic_irq(unsigned int irq)
{
+
+     move_irq(irq);
    if ((irq_desc[irq].status & (IRQ_PENDING | IRQ_DISABLED))
        == (IRQ_PENDING | IRQ_DISABLED))
        mask_IO_APIC_irq(irq);
@@ -1341,26 +1376,10 @@ static unsigned int startup_level_ioapic
static void end_level_ioapic_irq (unsigned int irq)
{
+
+     move_irq(irq);
    ack_APIC_irq();
}

-static void set_ioapic_affinity_irq(unsigned int irq, cpumask_t mask)
- {
-     unsigned long flags;
-     unsigned int dest;
-
-     dest = cpu_mask_to_apicid(mask);

```

Linux-Kernel: Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to -mm

```
-
- /*
-  * Only the high 8 bits are valid.
-  */
-   dest = SET_APIC_LOGICAL_ID(dest);
-
-   spin_lock_irqsave(&ioapic_lock, flags);
-   __DO_ACTION(1, = dest, )
-   spin_unlock_irqrestore(&ioapic_lock, flags);
-}
-
#ifdef CONFIG_PCI_MSI
static unsigned int startup_edge_ioapic_vector(unsigned int vector)
{
@@ -1373,6 +1392,7 @@ static void ack_edge_ioapic_vector(unsigned
{
    int irq = vector_to_irq(vector);

+   move_native_irq(vector);
    ack_edge_ioapic_irq(irq);
}

@@ -1387,6 +1407,7 @@ static void end_level_ioapic_vector (unsigned
{
    int irq = vector_to_irq(vector);

+   move_native_irq(vector);
    end_level_ioapic_irq(irq);
}

@@ -1409,6 +1430,7 @@ static void set_ioapic_affinity_vector (
{
    int irq = vector_to_irq(vector);

+   set_native_irq_info(vector, cpu_mask);
    set_ioapic_affinity_irq(irq, cpu_mask);
}
#endif
@@ -1979,6 +2001,7 @@ int io_apic_set_pci_routing (int ioapic,
spin_lock_irqsave(&ioapic_lock, flags);
io_apic_write(ioapic, 0x11+2*pin, *(((int *)&entry)+1));
io_apic_write(ioapic, 0x10+2*pin, *(((int *)&entry)+0));
+   set_native_irq_info(use_pci_vector() ? entry.vector : irq, TARGET_CPUS);
spin_unlock_irqrestore(&ioapic_lock, flags);

return 0;
diff -puN arch/x86_64/Kconfig~fix_irq_affinity arch/x86_64/Kconfig
--- linux-2.6.12-rc2-mm3/arch/x86_64/Kconfig~fix_irq_affinity 2005-04-28 23:51:28.000000000 -0700
+++ linux-2.6.12-rc2-mm3-araj/arch/x86_64/Kconfig 2005-04-28 23:51:28.000000000 -0700
@@ -409,6 +409,11 @@ config GENERIC_IRQ_PROBE
bool
default y

+config GENERIC_PENDING_IRQ
+bool
+depends on GENERIC_HARDIRQS && SMP
+default y
+
menu "Power management options"

source kernel/power/Kconfig
diff -puN arch/i386/Kconfig~fix_irq_affinity arch/i386/Kconfig
```

Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to 14mm tree

Linux-Kernel: Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to -mm

```
--- linux-2.6.12-rc2-mm3/arch/i386/Kconfig~fix_irq_affinity      2005-04-28 23:51:28.000000000 -07
+++ linux-2.6.12-rc2-mm3-araj/arch/i386/Kconfig 2005-04-28 23:51:28.000000000 -0700
@@ -1302,6 +1302,11 @@ config GENERIC_IRQ_PROBE
    bool
    default y

+config GENERIC_PENDING_IRQ
+   bool
+   depends on GENERIC_HARDIRQS && SMP
+   default y
+
+config X86_SMP
+   bool
+   depends on SMP && !X86_VOYAGER
diff -puN arch/ia64/Kconfig~fix_irq_affinity arch/ia64/Kconfig
--- linux-2.6.12-rc2-mm3/arch/ia64/Kconfig~fix_irq_affinity      2005-04-28 23:51:28.000000000 -07
+++ linux-2.6.12-rc2-mm3-araj/arch/ia64/Kconfig 2005-04-28 23:51:28.000000000 -0700
@@ -411,6 +411,11 @@ config GENERIC_IRQ_PROBE
    bool
    default y

+config GENERIC_PENDING_IRQ
+   bool
+   depends on GENERIC_HARDIRQS && SMP
+   default y
+
+source "arch/ia64/hp/sim/Kconfig"

+source "arch/ia64/oprofile/Kconfig"
diff -puN kernel/irq/proc.c~fix_irq_affinity kernel/irq/proc.c
--- linux-2.6.12-rc2-mm3/kernel/irq/proc.c~fix_irq_affinity      2005-04-28 23:51:28.000000000 -07
+++ linux-2.6.12-rc2-mm3-araj/kernel/irq/proc.c 2005-04-29 00:11:53.000000000 -0700
@@ -19,12 +19,27 @@ static struct proc_dir_entry *root_irq_d
    */
    static struct proc_dir_entry *smp_affinity_entry[NR_IRQS];

-void __attribute__((weak))
-proc_set_irq_affinity(unsigned int irq, cpumask_t mask_val)
+#ifdef CONFIG_GENERIC_PENDING_IRQ
+void proc_set_irq_affinity(unsigned int irq, cpumask_t mask_val)
+{
+   irq_desc_t      *desc = irq_descp(irq);
+   unsigned long flags;
+
+   /*
+    * Save these away for later use. Re-program when the
+    * interrupt is pending
+    */
+   spin_lock_irqsave(&desc->lock, flags);
+   pending_irq_cpumask[irq] = mask_val;
+   spin_unlock_irqrestore(&desc->lock, flags);
+}
+#else
+void proc_set_irq_affinity(unsigned int irq, cpumask_t mask_val)
+{
+   irq_affinity[irq] = mask_val;
+   irq_desc[irq].handler->set_affinity(irq, mask_val);
+}
+#endif

static int irq_affinity_read_proc(char *page, char **start, off_t off,
                                int count, int *eof, void *data)
```

Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to 12mm tree

Linux-Kernel: Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to -mm

```
diff -puN include/asm-ia64/hw_irq.h~fix_irq_affinity include/asm-ia64/hw_irq.h
--- linux-2.6.12-rc2-mm3/include/asm-ia64/hw_irq.h~fix_irq_affinity      2005-04-28 23:51:28.00000
+++ linux-2.6.12-rc2-mm3-araj/include/asm-ia64/hw_irq.h 2005-04-28 23:51:28.000000000 -0700
@@ -117,13 +117,6 @@ __ia64_local_vector_to_irq (ia64_vector
 * and to obtain the irq descriptor for a given irq number.
 */

-/* Return a pointer to the irq descriptor for IRQ. */
-static inline irq_desc_t *
-irq_descp (int irq)
-{
-    return irq_desc + irq;
-}
-
/* Extract the IA-64 vector that corresponds to IRQ. */
static inline ia64_vector
irq_to_vector (int irq)
diff -puN include/asm-ia64/irq.h~fix_irq_affinity include/asm-ia64/irq.h
--- linux-2.6.12-rc2-mm3/include/asm-ia64/irq.h~fix_irq_affinity      2005-04-29 07:17:43.00000
+++ linux-2.6.12-rc2-mm3-araj/include/asm-ia64/irq.h 2005-04-29 07:17:59.000000000 -0700
@@ -30,12 +30,6 @@ extern void disable_irq_nosync (unsigned
extern void enable_irq (unsigned int);
extern void set_irq_affinity_info (unsigned int irq, int dest, int redir);

-#ifdef CONFIG_SMP
-extern void move_irq(int irq);
-#else
-#define move_irq(irq)
-#endif
-
struct irqaction;
struct pt_regs;
int handle_IRQ_event(unsigned int, struct pt_regs *, struct irqaction *);
-
-
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/
```

Re: patch x86-x86_64-deferred-handling-of-writes-to-proc-irq-xx-smp_affinity.patch added to 13mm tree