

[PATCH] [7/34] x86_64: safe_apic_wait_icr_idle – x86_64

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-04/msg12090.html>

- *From:* Andi Kleen <ak@xxxxxxx>
 - *Date:* Mon, 30 Apr 2007 17:49:39 +0200 (CEST)
-

From: Fernando Luis VazquezCao <fernando@xxxxxxxxxxxxxxxx>

apic_wait_icr_idle looks like this:

```
static __inline__ void apic_wait_icr_idle(void)
{
while (apic_read(APIC_ICR) & APIC_ICR_BUSY)
cpu_relax();
}
```

The busy loop in this function would not be problematic if the corresponding status bit in the ICR were always updated, but that does not seem to be the case under certain crash scenarios. Kdump uses an IPI to stop the other CPUs in the event of a crash, but when any of the other CPUs are locked-up inside the NMI handler the CPU that sends the IPI will end up looping forever in the ICR check, effectively hard-locking the whole system.

Quoting from Intel's "MultiProcessor Specification" (Version 1.4), B-3:

"A local APIC unit indicates successful dispatch of an IPI by resetting the Delivery Status bit in the Interrupt Command Register (ICR). The operating system polls the delivery status bit after sending an INIT or STARTUP IPI until the command has been dispatched.

A period of 20 microseconds should be sufficient for IPI dispatch to complete under normal operating conditions. If the IPI is not successfully dispatched, the operating system can abort the command. Alternatively, the operating system can retry the IPI by writing the lower 32-bit double word of the ICR. This "time-out" mechanism can be implemented through an external interrupt, if interrupts are enabled on the processor, or through execution of an instruction or time-stamp counter spin loop."

Intel's documentation suggests the implementation of a time-out mechanism, which, by the way, is already being open-coded in some parts

of the kernel that tinker with ICR.

Create a apic_wait_icr_idle replacement that implements the time-out mechanism and that can be used to solve the aforementioned problem.

AK: moved both functions out of line
AK: Added improved loop from Keith Owens

Signed-off-by: Fernando Luis Vazquez Cao <fernando@xxxxxxxxxxxx>
Signed-off-by: Andi Kleen <ak@xxxxxxx>

arch/x86_64/kernel/apic.c | 22 ++++++
include/asm-x86_64/apic.h | 8 +++-----
2 files changed, 25 insertions(+), 5 deletions(-)

Index: linux/include/asm-x86_64/apic.h

```
=====
--- linux.orig/include/asm-x86_64/apic.h
+++ linux/include/asm-x86_64/apic.h
@@ -2,6 +2,7 @@
#define __ASM_APIC_H

#include <linux/pm.h>
+#include <linux/delay.h>
#include <asm/fixmap.h>
#include <asm/apicdef.h>
#include <asm/system.h>
@@ -47,11 +48,8 @@ static __inline unsigned int apic_read(u
return *((volatile unsigned int *) (APIC_BASE+reg));
}

-static __inline__ void apic_wait_icr_idle(void)
-{
- while (apic_read( APIC_ICR ) & APIC_ICR_BUSY)
- cpu_relax();
-}
+extern void apic_wait_icr_idle(void);
+extern unsigned int safe_apic_wait_icr_idle(void);
```

static inline void ack_APIC_irq(void)

```
{
Index: linux/arch/x86_64/kernel/apic.c
=====
--- linux.orig/arch/x86_64/kernel/apic.c
+++ linux/arch/x86_64/kernel/apic.c
@@ -68,6 +68,28 @@ int using_apic_timer __read_mostly = 0;

static void apic_pm_activate(void);
```

```
+void apic_wait_icr_idle(void)
+{
+ while (apic_read(APIC_ICR) & APIC_ICR_BUSY)
+ cpu_relax();
+}
+
+unsigned int safe_apic_wait_icr_idle(void)
+{
+ unsigned int send_status;
+ int timeout;
+
+ timeout = 0;
+ do {
+ send_status = apic_read(APIC_ICR) & APIC_ICR_BUSY;
+ if (!send_status)
+ break;
+ udelay(100);
+ } while (timeout++ < 1000);
+
+ return send_status;
+}
+
+void enable_NMI_through_LVT0 (void * dummy)
+{
+ unsigned int v;
```

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