

[PATCH] Dynamic tick for x86 version 050602-1

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-06/0461.html>

From: Tony Lindgren (tony_at_atomide.com)

Date: 06/02/05

Date: Wed, 1 Jun 2005 18:36:41 -0700
To: linux-kernel@vger.kernel.org

Hi all,

Here's an updated version of the dynamic tick patch.

It's mostly clean-up and it's now using the standard `monotonic_clock()` functions as suggested by John Stultz.

Please let me know of any issues with the patch. I'll continue to do more clean-up on it, but I think the basic functionality is done.

Thomas, where do you have the latest version of your ACPI idle patch? I'd like to add that to the dyn-tick page as well.

Older patches and some related links are at:

<http://muru.com/linux/dyntick/>

Regards,

Tony

Index: `linux-dev/arch/i386/Kconfig`

```
-----  
--- linux-dev.orig/arch/i386/Kconfig 2005-06-01 17:51:36.000000000 -0700  
+++ linux-dev/arch/i386/Kconfig 2005-06-01 17:54:32.000000000 -0700  
@@ -458,6 +458,26 @@ config HPET_EMULATE_RTC  
    bool "Provide RTC interrupt"  
    depends on HPET_TIMER && RTC=y
```

```
+config NO_IDLE_HZ  
+ bool "Dynamic Tick Timer - Skip timer ticks during idle"  
+ help  
+ This option enables support for skipping timer ticks when the  
+ processor is idle. During system load, timer is continuous.  
+ This option saves power, as it allows the system to stay in
```

Linux-Kernel: [PATCH] Dynamic tick for x86 version 050602-1

+ idle mode longer. Currently supported timers are ACPI PM
+ timer, local APIC timer, and TSC timer. HPET timer is currently
+ not supported.

+

+config DYN_TICK_USE_APIC

+ bool "Use APIC timer instead of PIT timer"

+ help

+ This option enables using APIC timer interrupt if your hardware
+ supports it. APIC timer allows longer sleep periods compared
+ to PIT timer. Note that on some hardware disabling PIT timer
+ also disables APIC timer interrupts, and system won't run
+ properly. Symptoms include slow system boot, and time running
+ slow. If unsure, don't enable this option.

+

config SMP

bool "Symmetric multi-processing support"

---help---

Index: linux-dev/arch/i386/kernel/Makefile

--- linux-dev.orig/arch/i386/kernel/Makefile 2005-06-01 17:51:36.000000000 -0700

+++ linux-dev/arch/i386/kernel/Makefile 2005-06-01 17:54:32.000000000 -0700

@@ -31,6 +31,7 @@ obj-\$(CONFIG_MODULES) += module.o

obj-y += sysenter.o vsyscall.o

obj-\$(CONFIG_ACPI_SRAT) += srat.o

obj-\$(CONFIG_HPET_TIMER) += time_hpet.o

+obj-\$(CONFIG_NO_IDLE_HZ) += dyn-tick.o

obj-\$(CONFIG_EFI) += efi.o efi_stub.o

obj-\$(CONFIG_EARLY_PRINTK) += early_printk.o

Index: linux-dev/arch/i386/kernel/apic.c

--- linux-dev.orig/arch/i386/kernel/apic.c 2005-06-01 17:51:36.000000000 -0700

+++ linux-dev/arch/i386/kernel/apic.c 2005-06-01 17:54:32.000000000 -0700

@@ -26,6 +26,7 @@

#include <linux/mc146818rtc.h>

#include <linux/kernel_stat.h>

#include <linux/sysdev.h>

+#include <linux/dyn-tick.h>

#include <asm/atomic.h>

#include <asm/smp.h>

@@ -909,6 +910,8 @@ void (*wait_timer_tick)(void) __initdata

#define APIC_DIVISOR 16

+static u32 apic_timer_val;

+

static void __setup_APIC_LVTT(unsigned int clocks)

{

unsigned int lvtt_value, tmp_value, ver;

@@ -927,7 +930,15 @@ static void __setup_APIC_LVTT(unsigned int i

```

& ~(APIC_TDR_DIV_1 | APIC_TDR_DIV_TMBASE))
| APIC_TDR_DIV_16);

```

```

- apic_write_around(APIC_TMICT, clocks/APIC_DIVISOR);
+ apic_timer_val = clocks/APIC_DIVISOR;
+
+ #ifdef CONFIG_NO_IDLE_HZ
+ /* Local APIC timer is 24-bit */
+ if (apic_timer_val)
+ dyn_tick->max_skip = 0xfffff / apic_timer_val;
+ #endif
+
+ apic_write_around(APIC_TMICT, apic_timer_val);
}

static void __init setup_APIC_timer(unsigned int clocks)
@@ -1040,6 +1051,13 @@ void __init setup_boot_APIC_clock(void)
    */
    setup_APIC_timer(calibration_result);

+ #ifdef CONFIG_NO_IDLE_HZ
+ if (calibration_result)
+ dyn_tick->state |= DYN_TICK_USE_APIC;
+ else
+ printk(KERN_INFO "dyn-tick: Cannot use local APIC\n");
+ #endif
+
+ local_irq_enable();
}

@@ -1068,6 +1086,18 @@ void enable_APIC_timer(void)
}
}

+ #if defined(CONFIG_NO_IDLE_HZ)
+ void reprogram_apic_timer(unsigned int count)
+ {
+ unsigned long flags;
+
+ count *= apic_timer_val;
+ local_irq_save(flags);
+ apic_write_around(APIC_TMICT, count);
+ local_irq_restore(flags);
+ }
+ #endif
+
+ /*
+  * the frequency of the profiling timer can be changed
+  * by writing a multiplier value into /proc/profile.
+  */
@@ -1160,6 +1190,7 @@ inline void smp_local_timer_interrupt(st

```

Linux-Kernel: [PATCH] Dynamic tick for x86 version 050602-1

```

fastcall void smp_apic_timer_interrupt(struct pt_regs *regs)
{
+ unsigned long seq;
    int cpu = smp_processor_id();

    /*
@@ -1178,6 +1209,23 @@ fastcall void smp_apic_timer_interrupt(s
    * interrupt lock, which is the WrongThing (tm) to do.
    */
    irq_enter();
+
+ #ifdef CONFIG_NO_IDLE_HZ
+ /*
+ * Check if we need to wake up PIT interrupt handler.
+ * Otherwise just wake up local APIC timer.
+ */
+ do {
+ seq = read_seqbegin(&xtime_lock);
+ if (dyn_tick->state & (DYN_TICK_ENABLED | DYN_TICK_SKIPPING)) {
+ if (dyn_tick->skip_cpu == cpu && dyn_tick->skip > DYN_TICK_MIN_SKIP)
+ dyn_tick->interrupt(99, NULL, regs);
+ else
+ reprogram_apic_timer(1);
+ }
+ } while (read_seqretry(&xtime_lock, seq));
+ #endif
+
    smp_local_timer_interrupt(regs);
    irq_exit();
}

```

Index: linux-dev/arch/i386/kernel/dyn-tick.c

```

=====
--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-dev/arch/i386/kernel/dyn-tick.c 2005-06-01 17:58:43.000000000 -0700
@@ -0,0 +1,45 @@
+/*
+ * linux/arch/i386/kernel/dyn-tick.c
+ *
+ * Copyright (C) 2004 Nokia Corporation
+ * Written by Tony Lindgen <tony@atomide.com> and
+ * Tuukka Tikkanen <tuukka.tikkanen@elektrobit.com>
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
+ */
+
+ #include <linux/version.h>
+ #include <linux/config.h>
+ #include <linux/kernel.h>
+ #include <linux/init.h>

```

```

+#include <linux/module.h>
+#include <linux/dyn-tick.h>
+
+void arch_reprogram_timer(void)
+{
+ if (cpu_has_local_apic()) {
+  disable_pit_timer();
+  if (dyn_tick->state & DYN_TICK_TIMER_INT)
+  reprogram_apic_timer(dyn_tick->skip);
+ } else {
+  if (dyn_tick->state & DYN_TICK_TIMER_INT)
+  reprogram_pit_timer(dyn_tick->skip);
+  else
+  disable_pit_timer();
+ }
+}
+
+static struct dyn_tick_timer arch_dyn_tick_timer = {
+ .arch_reprogram_timer = &arch_reprogram_timer,
+};
+
+int __init dyn_tick_init(void)
+{
+ arch_dyn_tick_timer.arch_init = dyn_tick_arch_init;
+ dyn_tick_register(&arch_dyn_tick_timer);
+
+ return 0;
+}
+arch_initcall(dyn_tick_init);
Index: linux-dev/include/asm-i386/dyn-tick.h

```

```

----- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-dev/include/asm-i386/dyn-tick.h 2005-06-01 17:58:55.000000000 -0700
@@ -0,0 +1,37 @@
+/*
+ * linux/include/asm-i386/dyn-tick.h
+ *
+ * Copyright (C) 2004 Nokia Corporation
+ * Written by Tony Lindgen <tony@atomide.com> and
+ * Tuukka Tikkanen <tuukka.tikkanen@elektrobit.com>
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
+ */
+
+extern int dyn_tick_arch_init(void);
+extern void disable_pit_timer(void);
+extern void reprogram_pit_timer(int jiffies_to_skip);
+extern void reprogram_apic_timer(unsigned int count);
+extern void replace_timer_interrupt(void * new_handler);

```

```

+
+ #if defined(CONFIG_NO_IDLE_HZ) && defined(CONFIG_X86_LOCAL_APIC)
+ extern void reprogram_apic_timer(unsigned int count);
+ #else
+ void reprogram_apic_timer(unsigned int count) { }
+ #endif
+
+ #undef DEBUG
+ #ifdef DEBUG
+ #define dbg_dyn_tick_irq() {if (skipped && skipped < dyn_tick->skip) \
+ printk("%u/%li ", skipped, dyn_tick->skip);}
+ #else
+ #define dbg_dyn_tick_irq() { }
+ #endif
+
+ #if defined(CONFIG_DYN_TICK_USE_APIC) && (defined(CONFIG_SMP) ||
+ defined(CONFIG_X86_UP_APIC))
+ #define cpu_has_local_apic() (dyn_tick->state & DYN_TICK_USE_APIC)
+ #else
+ #define cpu_has_local_apic() 0
+ #endif
Index: linux-dev/arch/i386/kernel/irq.c
=====
--- linux-dev.orig/arch/i386/kernel/irq.c 2005-06-01 17:51:36.000000000 -0700
+++ linux-dev/arch/i386/kernel/irq.c 2005-06-01 17:54:32.000000000 -0700
@@ -15,6 +15,7 @@
#include <linux/seq_file.h>
#include <linux/interrupt.h>
#include <linux/kernel_stat.h>
#include <linux/dyn-tick.h>

DEFINE_PER_CPU(irq_cpustat_t, irq_stat) ____cacheline_maxaligned_in_smp;
EXPORT_PER_CPU_SYMBOL(irq_stat);
@@ -102,6 +103,12 @@ fastcall unsigned int do_IRQ(struct pt_r
);
} else
#endif
+
+ #ifdef CONFIG_NO_IDLE_HZ
+ if (dyn_tick->state & (DYN_TICK_ENABLED | DYN_TICK_SKIPPING) && irq != 0)
+ dyn_tick->interrupt(irq, NULL, regs);
+ #endif
+
+ __do_IRQ(irq, regs);

irq_exit();
Index: linux-dev/arch/i386/kernel/process.c
=====
--- linux-dev.orig/arch/i386/kernel/process.c 2005-06-01 17:51:36.000000000 -0700
+++ linux-dev/arch/i386/kernel/process.c 2005-06-01 17:54:32.000000000 -0700
@@ -37,6 +37,7 @@

```

Linux-Kernel: [PATCH] Dynamic tick for x86 version 050602-1

```
#include <linux/kallsyms.h>
#include <linux/ptrace.h>
#include <linux/random.h>
+#include <linux/dyn-tick.h>

#include <asm/uaccess.h>
#include <asm/pgtable.h>
@@ -160,6 +161,10 @@ void cpu_idle (void)
        if (!idle)
            idle = default_idle;

+#ifdef CONFIG_NO_IDLE_HZ
+ dyn_tick_reprogram_timer();
+#endif
+
        __get_cpu_var(irq_stat).idle_timestamp = jiffies;
        idle();
    }
```

Index: linux-dev/arch/i386/kernel/time.c

```
----- linux-dev.orig/arch/i386/kernel/time.c 2005-06-01 17:51:36.000000000 -0700
+++ linux-dev/arch/i386/kernel/time.c 2005-06-01 17:59:25.000000000 -0700
@@ -46,6 +46,7 @@
#include <linux/bcd.h>
#include <linux/efi.h>
#include <linux/mca.h>
+#include <linux/dyn-tick.h>

#include <asm/io.h>
#include <asm/smp.h>
@@ -308,6 +309,51 @@ irqreturn_t timer_interrupt(int irq, voi
    return IRQ_HANDLED;
}

+#ifdef CONFIG_NO_IDLE_HZ
+static unsigned long long last_tick;
+
+/*
+ * This interrupt handler updates the time based on number of jiffies skipped
+ * It would be somewhat more optimized to have a customa handler in each timer
+ * using hardware ticks instead of nanoseconds. Note that CONFIG_NO_IDLE_HZ
+ * currently disables timer fallback on skipped jiffies.
+ */
+irqreturn_t dyn_tick_timer_interrupt(int irq, void *dev_id, struct pt_regs *regs)
+{
+    unsigned long flags;
+    volatile unsigned long long now;
+    unsigned int skipped = 0;
+
+    if (dyn_tick->state & DYN_TICK_DEBUG) {
+    if (irq == 0)
```

```

+ printk(".");
+ else
+ printk("%i ", irq);
+ }
+
+ write_seqlock_irqsave(&xtime_lock, flags);
+ now = cur_timer->monotonic_clock();
+ while (now - last_tick >= NS_TICK_LEN) {
+ last_tick += NS_TICK_LEN;
+ cur_timer->mark_offset();
+ do_timer_interrupt(irq, NULL, regs);
+ skipped++;
+ }
+ if (dyn_tick->state & (DYN_TICK_ENABLED | DYN_TICK_SKIPPING)) {
+ dbg_dyn_tick_irq();
+ dyn_tick->skip = 1;
+ if (cpu_has_local_apic())
+ reprogram_apic_timer(dyn_tick->skip);
+ reprogram_pit_timer(dyn_tick->skip);
+ dyn_tick->state |= DYN_TICK_ENABLED;
+ dyn_tick->state &= ~DYN_TICK_SKIPPING;
+ }
+ write_sequnlock_irqrestore(&xtime_lock, flags);
+
+ return IRQ_HANDLED;
+}
+
+/* CONFIG_NO_IDLE_HZ */
+
+/* not static: needed by APM */
+unsigned long get_cmos_time(void)
+{
+@@ -416,7 +462,7 @@ static struct sysdev_class timer_sysclas
+
+/* XXX this driverfs stuff should probably go elsewhere later -john */
+static struct sys_device device_timer = {
+struct sys_device device_timer = {
+    .id = 0,
+    .cls = &timer_sysclass,
+};
+@@ -452,6 +498,32 @@ static void __init hpet_time_init(void)
+}
+
+endif
+
+ifndef CONFIG_NO_IDLE_HZ
+
+int __init dyn_tick_arch_init(void)
+{
+    unsigned long flags;
+
+    write_seqlock_irqsave(&xtime_lock, flags);

```


Linux-Kernel: [PATCH] Dynamic tick for x86 version 050602-1

```
@@ -184,6 +185,7 @@ static void mark_offset_pmtmr(void)
    first_run = 0;
    offset_delay = 0;
}
+#endif
}
```

Index: linux-dev/arch/i386/kernel/timers/timer_tsc.c

```
----- linux-dev.orig/arch/i386/kernel/timers/timer_tsc.c 2005-06-01 17:51:36.000000000 -0700
+++ linux-dev/arch/i386/kernel/timers/timer_tsc.c 2005-06-01 17:54:32.000000000 -0700
```

```
@@ -348,6 +348,7 @@ static void mark_offset_tsc(void)

    rdtsc(last_tsc_low, last_tsc_high);

+#ifndef CONFIG_NO_IDLE_HZ
    spin_lock(&i8253_lock);
    outb_p(0x00, PIT_MODE); /* latch the count ASAP */

@@ -415,11 +416,14 @@ static void mark_offset_tsc(void)
    cpufreq_delayed_get();
} else
    lost_count = 0;
+#endif
+
    /* update the monotonic base value */
    this_offset = ((unsigned long long)last_tsc_high<<32)|last_tsc_low;
    monotonic_base += cycles_2_ns(this_offset - last_offset);
    write_sequnlock(&monotonic_lock);

+#ifndef CONFIG_NO_IDLE_HZ
    /* calculate delay_at_last_interrupt */
    count = ((LATCH-1) - count) * TICK_SIZE;
    delay_at_last_interrupt = (count + LATCH/2) / LATCH;
@@ -430,6 +434,7 @@ static void mark_offset_tsc(void)
    */
    if (lost && abs(delay - delay_at_last_interrupt) > (900000/HZ))
        jiffies_64++;
+#endif
}
```

```
static int __init init_tsc(char* override)
```

Index: linux-dev/arch/i386/mach-default/setup.c

```
----- linux-dev.orig/arch/i386/mach-default/setup.c 2005-06-01 17:51:36.000000000 -0700
+++ linux-dev/arch/i386/mach-default/setup.c 2005-06-01 17:54:32.000000000 -0700
```

```
@@ -85,6 +85,22 @@ void __init time_init_hook(void)
    setup_irq(0, &irq0);
}
```

```

+/**
+ * replace_timer_interrupt – allow replacing timer interrupt handler
+ *
+ * Description:
+ * Can be used to replace timer interrupt handler with a more optimized
+ * handler. Used for enabling and disabling of CONFIG_NO_IDLE_HZ.
+ */
+void replace_timer_interrupt(void * new_handler)
+{
+ unsigned long flags;
+
+ write_seqlock_irqsave(&xtime_lock, flags);
+ irq0.handler = new_handler;
+ write_sequnlock_irqrestore(&xtime_lock, flags);
+}
+
+#ifdef CONFIG_MCA
+/**
+ * mca_nmi_hook – hook into MCA specific NMI chain
Index: linux-dev/include/linux/dyn-tick.h
=====
--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-dev/include/linux/dyn-tick.h 2005-06-01 17:54:33.000000000 -0700
@@ -0,0 +1,62 @@
+/**
+ * linux/include/linux/dyn-tick.h
+ *
+ * Copyright (C) 2004 Nokia Corporation
+ * Written by Tony Lindgen <tony@atomide.com> and
+ * Tuukka Tikkanen <tuukka.tikkanen@elektrobit.com>
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
+ */
+
+#ifndef _DYN_TICK_TIMER_H
+#define _DYN_TICK_TIMER_H
+
+#include <linux/interrupt.h>
+
+#define DYN_TICK_DEBUG (1 << 31)
+#define DYN_TICK_TIMER_INT (1 << 4)
+#define DYN_TICK_USE_APIC (1 << 3)
+#define DYN_TICK_SKIPPING (1 << 2)
+#define DYN_TICK_ENABLED (1 << 1)
+#define DYN_TICK_SUITABLE (1 << 0)
+
+struct dyn_tick_state {
+ unsigned int state; /* Current state */
+ int skip_cpu; /* Skip handling processor */

```

Linux-Kernel: [PATCH] Dynamic tick for x86 version 050602-1

```
+ unsigned long skip; /* Ticks to skip */
+ unsigned int max_skip; /* Max number of ticks to skip */
+ unsigned long irq_skip_mask; /* Do not update time from these irqs */
+ irqreturn_t (*interrupt)(int, void *, struct pt_regs *);
+};
+
+struct dyn_tick_timer {
+ int (*arch_init) (void);
+ void (*arch_enable) (void);
+ void (*arch_disable) (void);
+ void (*arch_reprogram_timer) (void);
+};
+
+extern struct dyn_tick_state * dyn_tick;
+extern void dyn_tick_register(struct dyn_tick_timer * new_timer);
+
+#define NS_TICK_LEN ((1 * 1000000000)/HZ)
+#define DYN_TICK_MIN_SKIP 2
+
+#ifdef CONFIG_NO_IDLE_HZ
+
+extern unsigned long dyn_tick_reprogram_timer(void);
+
+#else
+
+#define arch_has_safe_halt() 0
+#define dyn_tick_reprogram_timer() {}
+
+#endif /* CONFIG_NO_IDLE_HZ */
+
+/* Pick up arch specific header */
+#include <asm/dyn-tick.h>
+
+#endif /* _DYN_TICK_TIMER_H */
```

Index: linux-dev/kernel/Makefile

```
-----
--- linux-dev.orig/kernel/Makefile 2005-06-01 17:51:36.000000000 -0700
+++ linux-dev/kernel/Makefile 2005-06-01 17:54:33.000000000 -0700
@@ -28,6 +28,7 @@ obj-$(CONFIG_KPROBES) += kprobes.o
obj-$(CONFIG_SYSFS) += ksysfs.o
obj-$(CONFIG_GENERIC_HARDIRQS) += irq/
obj-$(CONFIG_SECCOMP) += seccomp.o
+obj-$(CONFIG_NO_IDLE_HZ) += dyn-tick.o
```

```
ifneq ($(CONFIG_SCHED_NO_NO_OMIT_FRAME_POINTER),y)
# According to Alan Modra <alan@linuxcare.com.au>, the -fno-omit-frame-pointer is
Index: linux-dev/kernel/dyn-tick.c
```

```
-----
--- /dev/null 1970-01-01 00:00:00.000000000 +0000
+++ linux-dev/kernel/dyn-tick.c 2005-06-01 18:04:08.000000000 -0700
```

```

@@ -0,0 +1,235 @@
+/*
+ * linux/arch/i386/kernel/dyn-tick.c
+ *
+ * Beginnings of generic dynamic tick timer support
+ *
+ * Copyright (C) 2004 Nokia Corporation
+ * Written by Tony Lindgen <tony@atomide.com> and
+ * Tuukka Tikkanen <tuukka.tikkanen@elektrobit.com>
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
+ */
+
+#include <linux/version.h>
+#include <linux/config.h>
+#include <linux/kernel.h>
+#include <linux/init.h>
+#include <linux/module.h>
+#include <linux/sysdev.h>
+#include <linux/interrupt.h>
+#include <linux/cpumask.h>
+#include <linux/pm.h>
+#include <linux/dyn-tick.h>
+#include <asm/io.h>
+
+#include "io_ports.h"
+
+#define DYN_TICK_VERSION "050602-1"
+
+struct dyn_tick_state dyn_tick_state;
+struct dyn_tick_state * dyn_tick = &dyn_tick_state;
+struct dyn_tick_timer * dyn_tick_cfg;
+static cpumask_t dyn_cpu_map;
+
+/*
+ * Arch independent code needed to reprogram next timer interrupt.
+ * Gets called from cpu_idle() before entering idle loop. Note that
+ * we want to have all processors idle before reprogramming the
+ * next timer interrupt.
+ */
+unsigned long dyn_tick_reprogram_timer(void)
+{
+ int cpu;
+ unsigned long flags;
+ cpumask_t idle_cpus;
+ unsigned long next;
+
+ if (dyn_tick->state & DYN_TICK_DEBUG)
+ printk("i");

```

```

+
+ if (!(dyn_tick->state & DYN_TICK_ENABLED))
+ return 0;
+
+ /* Check if we are already skipping ticks and can idle other cpus */
+ if (dyn_tick->state & DYN_TICK_SKIPPING) {
+ reprogram_apic_timer(dyn_tick->skip);
+ return 0;
+ }
+
+ /* Check if we can start skipping ticks */
+ write_seqlock_irqsave(&xtime_lock, flags);
+ cpu = smp_processor_id();
+ cpu_set(cpu, dyn_cpu_map);
+ cpus_and(idle_cpus, dyn_cpu_map, cpu_online_map);
+ if (cpus_equal(idle_cpus, cpu_online_map)) {
+ next = next_timer_interrupt();
+ if (jiffies > next) {
+ //printk("Too late? next: %lu jiffies: %lu\n",
+ // next, jiffies);
+ dyn_tick->skip = 1;
+ } else
+ dyn_tick->skip = next_timer_interrupt() - jiffies;
+ if (dyn_tick->skip > DYN_TICK_MIN_SKIP) {
+ if (dyn_tick->skip > dyn_tick->max_skip)
+ dyn_tick->skip = dyn_tick->max_skip;
+ }
+ dyn_tick_cfg->arch_reprogram_timer();
+
+ dyn_tick->skip_cpu = cpu;
+ dyn_tick->state |= DYN_TICK_SKIPPING;
+ }
+ cpus_clear(dyn_cpu_map);
+ }
+ write_sequnlock_irqrestore(&xtime_lock, flags);
+
+ return dyn_tick->skip;
+}
+
+void __init dyn_tick_register(struct dyn_tick_timer * arch_timer)
+{
+ dyn_tick_cfg = arch_timer;
+ printk(KERN_INFO "dyn-tick: Registering dynamic tick timer v%s\n",
+ DYN_TICK_VERSION);
+}
+
+/* -----
+ * Sysfs interface
+ * -----
+ */

```

```

+
+extern struct sys_device device_timer;
+
+static ssize_t show_dyn_tick_state(struct sys_device *dev, char *buf)
+{
+ return sprintf(buf, "suitable:\t%i\n"
+ "enabled:\t%i\n"
+ "skipping:\t%i\n"
+ "using APIC:\t%i\n"
+ "int enabled:\t%i\n"
+ "debug:\t\t%i\n",
+ dyn_tick->state & DYN_TICK_SUITABLE,
+ (dyn_tick->state & DYN_TICK_ENABLED) >> 1,
+ (dyn_tick->state & DYN_TICK_SKIPPING) >> 2,
+ (dyn_tick->state & DYN_TICK_USE_APIC) >> 3,
+ (dyn_tick->state & DYN_TICK_TIMER_INT) >> 4,
+ (dyn_tick->state & DYN_TICK_DEBUG) >> 31);
+}
+
+static ssize_t set_dyn_tick_state(struct sys_device *dev, const char * buf,
+ size_t count)
+{
+ unsigned long flags;
+ unsigned int enable = simple_strtoul(buf, NULL, 2);
+
+ write_seqlock_irqsave(&xtime_lock, flags);
+ if (enable) {
+ if (dyn_tick_cfg->arch_enable)
+ dyn_tick_cfg->arch_enable();
+ dyn_tick->state |= DYN_TICK_ENABLED;
+ } else {
+ if (dyn_tick_cfg->arch_disable)
+ dyn_tick_cfg->arch_disable();
+ dyn_tick->state &= ~DYN_TICK_ENABLED;
+ }
+ write_sequnlock_irqrestore(&xtime_lock, flags);
+
+ return count;
+}
+
+static SYSDEV_ATTR(dyn_tick_state, 0644, show_dyn_tick_state,
+ set_dyn_tick_state);
+
+static ssize_t show_dyn_tick_int(struct sys_device *dev, char *buf)
+{
+ return sprintf(buf, "%i\n",
+ (dyn_tick->state & DYN_TICK_TIMER_INT) >> 4);
+}
+
+static ssize_t set_dyn_tick_int(struct sys_device *dev, const char * buf,
+ size_t count)

```

```

+{
+ unsigned long flags;
+ unsigned int enable = simple_strtoul(buf, NULL, 2);
+
+ write_seqlock_irqsave(&xtime_lock, flags);
+ if (enable)
+ dyn_tick->state |= DYN_TICK_TIMER_INT;
+ else
+ dyn_tick->state &= ~DYN_TICK_TIMER_INT;
+ write_sequnlock_irqrestore(&xtime_lock, flags);
+
+ return count;
+}
+
+static SYSDEV_ATTR(dyn_tick_int, 0644, show_dyn_tick_int, set_dyn_tick_int);
+
+static ssize_t show_dyn_tick_dbg(struct sys_device *dev, char *buf)
+{
+ return sprintf(buf, "%i\n",
+ (dyn_tick->state & DYN_TICK_DEBUG) >> 31);
+}
+
+static ssize_t set_dyn_tick_dbg(struct sys_device *dev, const char * buf,
+ size_t count)
+{
+ unsigned long flags;
+ unsigned int enable = simple_strtoul(buf, NULL, 2);
+
+ write_seqlock_irqsave(&xtime_lock, flags);
+ if (enable)
+ dyn_tick->state |= DYN_TICK_DEBUG;
+ else
+ dyn_tick->state &= ~DYN_TICK_DEBUG;
+ write_sequnlock_irqrestore(&xtime_lock, flags);
+
+ return count;
+}
+
+static SYSDEV_ATTR(dyn_tick_dbg, 0644, show_dyn_tick_dbg, set_dyn_tick_dbg);
+
+/*
+ * -----
+ * Init functions
+ * -----
+ */
+
+static int __init dyn_tick_early_init(void)
+{
+ dyn_tick->state |= DYN_TICK_TIMER_INT;
+ return 0;
+}

```

```

+
+subsys_initcall(dyn_tick_early_init);
+
+/*
+ * We need to initialize dynamic tick after calibrate delay
+ */
+static int __init dyn_tick_late_init(void)
+{
+ int ret = 0;
+
+
+ if (dyn_tick_cfg == NULL || dyn_tick_cfg->arch_init == NULL ||
+ !(dyn_tick->state & DYN_TICK_SUITABLE)) {
+ printk(KERN_ERR "dyn-tick: No suitable timer found\n");
+ return -ENODEV;
+ }
+
+
+ ret = dyn_tick_cfg->arch_init();
+ if (ret != 0) {
+ printk(KERN_ERR "dyn-tick: Init failed\n");
+ return -ENODEV;
+ }
+
+
+ ret = sysdev_create_file(&device_timer, &attr_dyn_tick_state);
+ ret = sysdev_create_file(&device_timer, &attr_dyn_tick_int);
+ ret = sysdev_create_file(&device_timer, &attr_dyn_tick_dbg);
+
+
+ printk(KERN_INFO "dyn-tick: Timer using dynamic tick\n");
+
+
+ return ret;
+}
+
+late_initcall(dyn_tick_late_init);
Index: linux-dev/arch/i386/kernel/timers/timer_pit.c

```

```

=====
--- linux-dev.orig/arch/i386/kernel/timers/timer_pit.c 2005-06-01 17:51:36.000000000 -0700
+++ linux-dev/arch/i386/kernel/timers/timer_pit.c 2005-06-01 17:54:33.000000000 -0700
@@ -149,6 +149,43 @@ static unsigned long get_offset_pit(void
     return count;
 }

```

```

+/*
+ * REVISIT: Looks like on P3 APIC timer keeps running if PIT mode
+ * is changed. On P4, changing PIT mode seems to kill
+ * APIC timer interrupts. Same thing with disabling PIT
+ * interrupt.
+ */
+void disable_pit_timer(void)
+{
+ extern spinlock_t i8253_lock;
+ unsigned long flags;
+ spin_lock_irqsave(&i8253_lock, flags);

```

Linux-Kernel: [PATCH] Dynamic tick for x86 version 050602-1

```
+ outb_p(0x32, PIT_MODE); /* binary, mode 1, LSB/MSB, ch 0 */
+ spin_unlock_irqrestore(&i8253_lock, flags);
+}
+
+/*
+ * Reprograms the next timer interrupt
+ * PIT timer reprogramming code taken from APM code.
+ * Note that PIT timer is a 16-bit timer, which allows max
+ * skip of only few seconds.
+ */
+void reprogram_pit_timer(int jiffies_to_skip)
+{
+ int skip;
+ extern spinlock_t i8253_lock;
+ unsigned long flags;
+
+ skip = jiffies_to_skip * LATCH;
+ if (skip > 0xffff)
+ skip = 0xffff;
+
+ spin_lock_irqsave(&i8253_lock, flags);
+ outb_p(0x34, PIT_MODE); /* binary, mode 2, LSB/MSB, ch 0 */
+ outb_p(skip & 0xff, PIT_CH0); /* LSB */
+ outb(skip >> 8, PIT_CH0); /* MSB */
+ spin_unlock_irqrestore(&i8253_lock, flags);
+}

/* tsc timer_opts struct */
struct timer_opts timer_pit = {
```

-

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/>