Subject: [PATCH][x86_64] IPI calltraces on x86_64
Posted by Kirill Korotaev on Mon, 17 Apr 2006 11:45:50 GMT
View Forum Message <> Reply to Message

[PATCH] IPI calltraces on x86_64

This patch makes x86_64 kernel to print calltraces on _all_
CPUs on Alt-SysRq-p and NMI LOCKUP.
This patch supplements the one made for i386
some time ago and included in -mm tree.
Since it proved its usefullness many times already
we did the same patch for x86_64.

Signed-Off-By: Pavel Emelianov <xemul@openvz.org>
Signed-Off-By: Kirill Korotaev <dev@openvz.org>

```
--- ./arch/x86_64/kernel/nmi.c.ve129 2006-03-20 08:53:29.000000000 +0300
+++ ./arch/x86_64/kernel/nmi.c 2006-04-05 17:35:49.000000000 +0400
@@ -522,6 +522,7 @@ static __kprobes int dummy_nmi_callback(
 }

 static nmi_callback_t nmi_callback = dummy_nmi_callback;
+static nmi_callback_t nmi_ipi_callback = dummy_nmi_callback;

 asmlinkage __kprobes void do_nmi(struct pt_regs * regs, long error_code)
 {
@@ -531,9 +532,21 @@ asmlinkage __kprobes void do_nmi(struct
  add_pda(__nmi_count,1);
  if (!rcu_dereference(nmi_callback)(regs, cpu))
   default_do_nmi(regs);
+
+ nmi_ipi_callback(regs, cpu);
  nmi_exit();
 }

+void set_nmi_ipi_callback(nmi_callback_t callback)
+{
+ nmi_ipi_callback = callback;
+}
+
+void unset_nmi_ipi_callback(void)
+{
+ nmi_ipi_callback = dummy_nmi_callback;
+}
+
 void set_nmi_callback(nmi_callback_t callback)
 {
  rcu_assign_pointer(nmi_callback, callback);
```

```
--- ./arch/x86_64/kernel/process.c.ve129 2006-04-05 16:52:55.000000000 +0400
+++ ./arch/x86_64/kernel/process.c 2006-04-05 18:13:16.000000000 +0400
@@ -345,6 +345,21 @@ void show_regs(struct pt_regs *regs)
  show_trace(&regs->rsp);
 }

+void smp_show_regs(struct pt_regs *regs, void *data)
+{
+ static DEFINE_SPINLOCK(show_regs_lock);
+
+ if (regs == NULL)
+  return;
+
+ bust_spinlocks(1);
+ spin_lock(&show_regs_lock);
+ printk("----------- IPI show regs -----------\n");
+ show_regs(regs);
+ spin_unlock(&show_regs_lock);
+ bust_spinlocks(0);
+}
+
 /*
  * Free current thread data structures etc..
  */
--- ./arch/x86_64/kernel/smp.c.ve129 2006-04-05 15:11:18.000000000 +0400
+++ ./arch/x86_64/kernel/smp.c 2006-04-05 18:15:30.000000000 +0400
@@ -28,6 +28,7 @@
 #include <asm/proto.h>
 #include <asm/apicdef.h>
 #include <asm/idle.h>
+#include <asm/nmi.h>

 /*
  * Smarter SMP flushing macros.
@@ -444,6 +445,84 @@ int smp_call_function (void (*func) (voi
  return 0;
 }

+static spinlock_t nmi_call_lock = SPIN_LOCK_UNLOCKED;
+static struct nmi_call_data_struct {
+ smp_nmi_function func;
+ void *info;
+ atomic_t started;
+ atomic_t finished;
+ cpumask_t cpus_called;
+ int wait;
+} *nmi_call_data;
+
```

```
+static int smp_nmi_callback(struct pt_regs * regs, int cpu)
+{
+ smp_nmi_function func;
+ void *info;
+ int wait;
+
+ func = nmi_call_data->func;
+ info = nmi_call_data->info;
+ wait = nmi_call_data->wait;
+ ack_APIC_irq();
+ /* prevent from calling func() multiple times */
+ if (cpu_test_and_set(cpu, nmi_call_data->cpus_called))
+  return 0;
+ /*
+  * notify initiating CPU that I've grabbed the data and am
+  * about to execute the function
+  */
+ mb();
+ atomic_inc(&nmi_call_data->started);
+ /* at this point the nmi_call_data structure is out of scope */
+ irq_enter();
+ func(regs, info);
+ irq_exit();
+ if (wait)
+  atomic_inc(&nmi_call_data->finished);
+
+ return 0;
+}
+
+int smp_nmi_call_function(smp_nmi_function func, void *info, int wait)
+{
+ struct nmi_call_data_struct data;
+ int cpus;
+
+ cpus = num_online_cpus() - 1;
+ if (!cpus)
+  return 0;
+
+ data.func = func;
+ data.info = info;
+ data.wait = wait;
+ atomic_set(&data.started, 0);
+ atomic_set(&data.finished, 0);
+ cpus_clear(data.cpus_called);
+ /* prevent this cpu from calling func if NMI happens */
+ cpu_set(smp_processor_id(), data.cpus_called);
+
+ if (!spin_trylock(&nmi_call_lock))
```

```
+  return -1;
+
+ nmi_call_data = &data;
+ set_nmi_ipi_callback(smp_nmi_callback);
+ mb();
+
+ /* Send a message to all other CPUs and wait for them to respond */
+ send_IPI_allbutself(APIC_DM_NMI);
+ while (atomic_read(&data.started) != cpus)
+  barrier();
+
+ unset_nmi_ipi_callback();
+ if (wait)
+  while (atomic_read(&data.finished) != cpus)
+   barrier();
+ spin_unlock(&nmi_call_lock);
+
+ return 0;
+}
+
 void smp_stop_cpu(void)
 {
  unsigned long flags;
--- ./arch/x86_64/kernel/traps.c.ve129 2006-04-05 16:52:55.000000000 +0400
+++ ./arch/x86_64/kernel/traps.c 2006-04-05 17:43:53.000000000 +0400
@@ -460,6 +460,7 @@ void __kprobes die_nmi(char *str, struct
  show_registers(regs);
  if (panic_on_timeout || panic_on_oops)
   panic("nmi watchdog");
+ smp_nmi_call_function(smp_show_regs, NULL, 1);
  printk("console shuts up ...\n");
  oops_end(flags);
  do_exit(SIGSEGV);
--- ./drivers/char/sysrq.c.ve129 2006-04-05 16:52:55.000000000 +0400
+++ ./drivers/char/sysrq.c 2006-04-05 17:55:06.000000000 +0400
@@ -178,7 +178,7 @@ static void sysrq_handle_showregs(int ke
  if (pt_regs)
   show_regs(pt_regs);
  bust_spinlocks(0);
-#ifdef __i386__
+#if defined(__i386__) || defined(__x86_64__)
  smp_nmi_call_function(smp_show_regs, NULL, 0);
 #endif
 }
--- ./include/asm-x86_64/nmi.h.ve129 2006-03-20 08:53:29.000000000 +0300
+++ ./include/asm-x86_64/nmi.h 2006-04-05 17:44:44.000000000 +0400
@@ -24,6 +24,9 @@ void set_nmi_callback(nmi_callback_t cal
  * Remove the handler previously set.
```

```
 */
 void unset_nmi_callback(void);
+
+void set_nmi_ipi_callback(nmi_callback_t callback);
+void unset_nmi_ipi_callback(void);

 #ifdef CONFIG_PM
```