Subject: [PATCH] Add ability to print calltraces tighter on i386
Posted by Pavel Emelianov on Wed, 08 Aug 2007 14:17:51 GMT
View Forum Message <> Reply to Message

When printing a BUG or OOPS report the longest part of it is
the calltrace, which sometimes (quite often) doesn't fit the
standard 25-lines display. This may become a bad news when the
system doesn't have a serial/net console and is completely
frozen so that the terminal scrolling doesn't work.

The information that hides from the developer is registers, the
top of the calltrace and information about the kernel and the
crashed process (uname). As our experience shows, seeing this
info is sometimes critical and having a short calltrace would
help a lot.

The proposal is to make a boot-option called "tight_trace", that
makes the calltrace show only the addresses in one line instead
of the symbol names one per line.

E.g. OOPSes of 50 lines occupy ~20 with this patch.

This is an example of how it will look for i386, but if this
will be found useful, I will make the patch for other arched
I can test it on (at least x86_64, ia64).

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
---

 arch/i386/kernel/traps.c |   29 +++++++++++++++++++++++++----
 1 files changed, 25 insertions(+), 4 deletions(-)

diff --git a/arch/i386/kernel/traps.c b/arch/i386/kernel/traps.c
index f469d20..8aa2b3b 100644
--- a/arch/i386/kernel/traps.c
+++ b/arch/i386/kernel/traps.c
@@ -186,6 +186,19 @@ void dump_trace(struct task_struct *task
 }
 EXPORT_SYMBOL(dump_trace);

+static int decode_oops = 1;
+
+static int __init setup_decode_oops(char *str)
+{
+ if (*str)
+  return 0;
+
+ decode_oops = 0;
+ return 1;
+}
+
+__setup("tight_trace", setup_decode_oops);
+
 static void
 print_trace_warning_symbol(void *data, char *msg, unsigned long symbol)
 {
@@ -209,8 +222,11 @@ static int print_trace_stack(void *data,
  */
 static void print_trace_address(void *data, unsigned long addr)
 {
- printk("%s [<%08lx>] ", (char *)data, addr);
- print_symbol("%s\n", addr);
+ if (decode_oops) {
+  printk("%s [<%08lx>] ", (char *)data, addr);
+  print_symbol("%s\n", addr);
+ } else
+  printk("[<%08lx>] ", addr);
  touch_nmi_watchdog();
 }

@@ -226,7 +242,9 @@ show_trace_log_lvl(struct task_struct *t
      unsigned long * stack, char *log_lvl)
 {
  dump_trace(task, regs, stack, &print_trace_ops, log_lvl);
- printk("%s =======================\n", log_lvl);
+ if (decode_oops)
+  printk("%s =======================", log_lvl);
+ printk("\n");
 }

 void show_trace(struct task_struct *task, struct pt_regs *regs,
@@ -256,7 +274,10 @@ static void show_stack_log_lvl(struct ta
    printk("\n%s       ", log_lvl);
   printk("%08lx ", *stack++);
  }
- printk("\n%sCall Trace:\n", log_lvl);
+ printk("\n%sCall Trace: ", log_lvl);
+ if (decode_oops)
+  printk("\n");
+
  show_trace_log_lvl(task, regs, esp, log_lvl);
  debug_show_held_locks(task);
 }
```