[NMI watchdog is available for x86 and x86-64 architectures]

Is your system locking up unpredictably? No keyboard activity, just
a frustrating complete hard lockup? Do you want to help us debugging
such lockups? If all yes then this document is definitely for you.

On many x86/x86-64 type hardware there is a feature that enables
us to generate 'watchdog NMI interrupts'. (NMI: Non Maskable Interrupt
which get executed even if the system is otherwise locked up hard).
This can be used to debug hard kernel lockups. By executing periodic
NMI interrupts, the kernel can monitor whether any CPU has locked up,
and print out debugging messages if so.

In order to use the NMI watchdog, you need to have APIC support in your
kernel. For SMP kernels, APIC support gets compiled in automatically. For
UP, enable either CONFIG_X86_UP_APIC (Processor type and features -> Local
APIC support on uniprocessors) or CONFIG_X86_UP_IOAPIC (Processor type and
features -> IO-APIC support on uniprocessors) in your kernel config.
CONFIG_X86_UP_APIC is for uniprocessor machines without an IO-APIC.
CONFIG_X86_UP_IOAPIC is for uniprocessor with an IO-APIC. [Note: certain
kernel debugging options, such as Kernel Stack Meter or Kernel Tracer,
may implicitly disable the NMI watchdog.]

For x86-64, the needed APIC is always compiled in.

Using local APIC (nmi_watchdog=2) needs the first performance register, so
you can't use it for other purposes (such as high precision performance
profiling.) However, at least oprofile and the perfctr driver disable the
local APIC NMI watchdog automatically.

To actually enable the NMI watchdog, use the 'nmi_watchdog=N' boot
parameter. Eg. the relevant lilo.conf entry:

        append="nmi_watchdog=1"

For SMP machines and UP machines with an IO-APIC use nmi_watchdog=1.
For UP machines without an IO-APIC use nmi_watchdog=2, this only works
for some processor types. If in doubt, boot with nmi_watchdog=1 and
check the NMI count in /proc/interrupts; if the count is zero then
reboot with nmi_watchdog=2 and check the NMI count. If it is still
zero then log a problem, you probably have a processor that needs to be
added to the nmi code.

A 'lockup' is the following scenario: if any CPU in the system does not
execute the period local timer interrupt for more than 5 seconds, then
the NMI handler generates an oops and kills the process. This
'controlled crash' (and the resulting kernel messages) can be used to
debug the lockup. Thus whenever the lockup happens, wait 5 seconds and
the oops will show up automatically. If the kernel produces no messages
then the system has crashed so hard (eg. hardware-wise) that either it
cannot even accept NMI interrupts, or the crash has made the kernel
unable to print messages.

Be aware that when using local APIC, the frequency of NMI interrupts
it generates, depends on the system load. The local APIC NMI watchdog,

lacking a better source, uses the "cycles unhalted" event. As you may guess it doesn't tick when the CPU is in the halted state (which happens when the system is idle), but if your system locks up on anything but the "hlt" processor instruction, the watchdog will trigger very soon as the "cycles unhalted" event will happen every clock tick. If it locks up on "hlt", then you are out of luck -- the event will not happen at all and the watchdog won't trigger. This is a shortcoming of the local APIC watchdog -- unfortunately there is no "clock ticks" event that would work all the time. The I/O APIC watchdog is driven externally and has no such shortcoming. But its NMI frequency is much higher, resulting in a more significant hit to the overall system performance.

On x86 nmi_watchdog is disabled by default so you have to enable it with a boot time parameter.

It's possible to disable the NMI watchdog in run-time by writing "0" to /proc/sys/kernel/nmi_watchdog. Writing "1" to the same file will re-enable the NMI watchdog. Notice that you still need to use "nmi_watchdog=" parameter at boot time.

NOTE: In kernels prior to 2.4.2-ac18 the NMI-oopser is enabled unconditionally on x86 SMP boxes.

[feel free to send bug reports, suggestions and patches to
  Ingo Molnar <mingo@redhat.com> or the Linux SMP mailing
  list at <linux-smp@vger.kernel.org>]