
Subject: rdmsr_on_cpu, wrmsr_on_cpu et al
Posted by [adobriyan](#) on Wed, 17 Jan 2007 13:20:36 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi, Dave!

There was OpenVZ specific bug rendering some cpufreq drivers unusable on SMP. In short, when cpufreq code thinks it confined itself to needed cpu by means of set_cpus_allowed() to execute rdmsr, some "virtual cpu" feature can migrate process to anywhere. This triggers bugons and does wrong things in general.

This got fixed by introducing rdmsr_on_cpu and wrmsr_on_cpu executing rdmsr and wrmsr on given physical cpu by means of smp_call_function_single().

So, understanding p4-clockmod is not the only driver that was affected, is something like this acceptable for start?

```
arch/i386/kernel/cpu/cpufreq/Makefile    | 2
arch/i386/kernel/cpu/cpufreq/msr-on-cpu.c | 71 +++++
arch/i386/kernel/cpu/cpufreq/msr-on-cpu.h | 9 +++
arch/i386/kernel/cpu/cpufreq/p4-clockmod.c | 31 ++-----
4 files changed, 88 insertions(+), 25 deletions(-)
```

```
--- a/arch/i386/kernel/cpu/cpufreq/Makefile
+++ b/arch/i386/kernel/cpu/cpufreq/Makefile
@@ -11,5 +11,5 @@ obj-$(CONFIG_X86_SPEEDSTEP_LIB) += spee
obj-$(CONFIG_X86_SPEEDSTEP_SMI) += speedstep-smi.o
obj-$(CONFIG_X86_ACPI_CPUFREQ) += acpi-cpufreq.o
obj-$(CONFIG_X86_SPEEDSTEP_CENTRINO) += speedstep-centrino.o
-obj-$(CONFIG_X86_P4_CLOCKMOD) += p4-clockmod.o
+obj-$(CONFIG_X86_P4_CLOCKMOD) += p4-clockmod.o msr-on-cpu.o
obj-$(CONFIG_X86_CPUFREQ_NFORCE2) += cpufreq-nforce2.o
--- /dev/null
+++ b/arch/i386/kernel/cpu/cpufreq/msr-on-cpu.c
@@ -0,0 +1,71 @@
+#include <linux/module.h>
+#include <linux/preempt.h>
+#include <linux/smp.h>
+#include <asm/msr.h>
+#include "msr-on-cpu.h"
+
+#ifdef CONFIG_SMP
+struct msr_info {
+    u32 msr_no;
+    u32 l, h;
+};
```

```

+
+static void __rdmsr_on_cpu(void *info)
+{
+ struct msr_info *rv = info;
+
+ rdmsr(rv->msr_no, rv->l, rv->h);
+}
+
+void rdmsr_on_cpu(unsigned int cpu, u32 msr_no, u32 *l, u32 *h)
+{
+ preempt_disable();
+ if (smp_processor_id() == cpu)
+ rdmsr(msr_no, *l, *h);
+ else {
+ struct msr_info rv;
+
+ rv.msr_no = msr_no;
+ smp_call_function_single(cpu, __rdmsr_on_cpu, &rv, 0, 1);
+ *l = rv.l;
+ *h = rv.h;
+ }
+ preempt_enable();
+}
+
+static void __wrmsr_on_cpu(void *info)
+{
+ struct msr_info *rv = info;
+
+ wrmsr(rv->msr_no, rv->l, rv->h);
+}
+
+void wrmsr_on_cpu(unsigned int cpu, u32 msr_no, u32 l, u32 h)
+{
+ preempt_disable();
+ if (smp_processor_id() == cpu)
+ wrmsr(msr_no, l, h);
+ else {
+ struct msr_info rv;
+
+ rv.msr_no = msr_no;
+ rv.l = l;
+ rv.h = h;
+ smp_call_function_single(cpu, __wrmsr_on_cpu, &rv, 0, 1);
+ }
+ preempt_enable();
+}
+
+void rdmsr_on_cpu(unsigned int cpu, u32 msr_no, u32 *l, u32 *h)

```



```

+ rdmsr_on_cpu(cpu, MSR_IA32_THERM_CONTROL, &l, &h);
  if (newstate == DC_DISABLE) {
    dprintk("CPU#%d disabling modulation\n", cpu);
- wrmsr(MSR_IA32_THERM_CONTROL, l & ~(1<<4), h);
+ wrmsr_on_cpu(cpu, MSR_IA32_THERM_CONTROL, l & ~(1<<4), h);
  } else {
    dprintk("CPU#%d setting duty cycle to %d%%\n",
      cpu, ((125 * newstate) / 10));
@@ -85,7 +86,7 @@ static int cpufreq_p4_setdc(unsigned int
  */
  l = (l & ~14);
  l = l | (1<<4) | ((newstate & 0x7)<<1);
- wrmsr(MSR_IA32_THERM_CONTROL, l, h);
+ wrmsr_on_cpu(cpu, MSR_IA32_THERM_CONTROL, l, h);
  }

  return 0;
@@ -112,7 +113,6 @@ static int cpufreq_p4_target(struct cpuf
{
  unsigned int  newstate = DC_RESV;
  struct cpufreq_freqs freqs;
- cpumask_t cpus_allowed;
  int i;

  if (cpufreq_frequency_table_target(policy, &p4clockmod_table[0], target_freq, relation,
&newstate))
@@ -133,17 +133,8 @@ static int cpufreq_p4_target(struct cpuf
/* run on each logical CPU, see section 13.15.3 of IA32 Intel Architecture Software
 * Developer's Manual, Volume 3
 */
- cpus_allowed = current->cpus_allowed;
-
- for_each_cpu_mask(i, policy->cpus) {
-  cpumask_t this_cpu = cpumask_of_cpu(i);
-
-  set_cpus_allowed(current, this_cpu);
-  BUG_ON(smp_processor_id() != i);
-
+ for_each_cpu_mask(i, policy->cpus)
  cpufreq_p4_setdc(i, p4clockmod_table[newstate].index);
- }
- set_cpus_allowed(current, cpus_allowed);

  /* notifiers */
  for_each_cpu_mask(i, policy->cpus) {
@@ -265,17 +256,9 @@ static int cpufreq_p4_cpu_exit(struct cp
static unsigned int cpufreq_p4_get(unsigned int cpu)

```

```
{
- cpumask_t cpus_allowed;
  u32 l, h;

- cpus_allowed = current->cpus_allowed;
-
- set_cpus_allowed(current, cpumask_of_cpu(cpu));
- BUG_ON(smp_processor_id() != cpu);
-
- rdmsr(MSR_IA32_THERM_CONTROL, l, h);
-
- set_cpus_allowed(current, cpus_allowed);
+ rdmsr_on_cpu(cpu, MSR_IA32_THERM_CONTROL, &l, &h);

  if (l & 0x10) {
    l = l >> 1;
  }
}
```
