
Subject: [PATCH 1/2] rdmsr_on_cpu, wrmsr_on_cpu
Posted by [adobriyan](#) on Wed, 31 Jan 2007 15:48:18 GMT
[View Forum Message](#) <> [Reply to Message](#)

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().

Dave Jones mentioned cpufreq might be not only user of rdmsr_on_cpu() and wrmsr_on_cpu(), so I'm putting them into arch/{i386,x86_64}/lib/.

Signed-off-by: Alexey Dobriyan <adobriyan@openvz.org>

```
arch/i386/lib/Makefile      | 2 +
arch/i386/lib/msr-on-cpu.c  | 70 +++++
arch/x86_64/lib/Makefile    | 2 -
arch/x86_64/lib/msr-on-cpu.c | 1
include/asm-i386/msr.h      | 3 +
include/asm-x86_64/msr.h    | 2 +
6 files changed, 79 insertions(+), 1 deletion(-)
```

--- a/arch/i386/lib/Makefile

+++ b/arch/i386/lib/Makefile

```
@ @ -7,3 +7,5 @ @ lib-y = checksum.o delay.o usercopy.o ge
bitops.o semaphore.o
```

```
lib-$(CONFIG_X86_USE_3DNOW) += mmx.o
```

```
+
```

```
+obj-y = msr-on-cpu.o
```

```
--- /dev/null
```

```
+++ b/arch/i386/lib/msr-on-cpu.c
```

```
@ @ -0,0 +1,70 @ @
```

```
+#include <linux/module.h>
```

```
+#include <linux/preempt.h>
```

```
+#include <linux/smp.h>
```

```
+#include <asm/msr.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();
+}
+
+#else

```

```

+void rdmsr_on_cpu(unsigned int cpu, u32 msr_no, u32 *l, u32 *h)
+{
+ rdmsr(msr_no, *l, *h);
+}
+
+void wrmsr_on_cpu(unsigned int cpu, u32 msr_no, u32 l, u32 h)
+{
+ wrmsr(msr_no, l, h);
+}
+#endif
+
+EXPORT_SYMBOL(rdmsr_on_cpu);
+EXPORT_SYMBOL(wrmsr_on_cpu);
--- a/arch/x86_64/lib/Makefile
+++ b/arch/x86_64/lib/Makefile
@@ -4,7 +4,7 @@ #

```

CFLAGS_csum-partial.o := -funroll-loops

```

-obj-y := io.o iomap_copy.o
+obj-y := io.o iomap_copy.o msr-on-cpu.o

```

```

lib-y := csum-partial.o csum-copy.o csum-wrappers.o delay.o \
  usercopy.o getuser.o putuser.o \
--- /dev/null
+++ b/arch/x86_64/lib/msr-on-cpu.c
@@ -0,0 +1 @@
+#include "../i386/lib/msr-on-cpu.c"
--- a/include/asm-i386/msr.h
+++ b/include/asm-i386/msr.h
@@ -83,6 +83,9 @@ #define rdpmc(counter,low,high) \
    : "c" (counter))
#endif /* !CONFIG_PARAVIRT */

```

```

+void rdmsr_on_cpu(unsigned int cpu, u32 msr_no, u32 *l, u32 *h);
+void wrmsr_on_cpu(unsigned int cpu, u32 msr_no, u32 l, u32 h);
+
+/* symbolic names for some interesting MSRs */
+/* Intel defined MSRs. */
+#define MSR_IA32_P5_MC_ADDR 0
--- a/include/asm-x86_64/msr.h
+++ b/include/asm-x86_64/msr.h
@@ -160,6 +160,8 @@ static inline unsigned int cpuid_edx(uns
#define MSR_IA32_UCODE_WRITE 0x79
#define MSR_IA32_UCODE_REV 0x8b

+void rdmsr_on_cpu(unsigned int cpu, u32 msr_no, u32 *l, u32 *h);
+void wrmsr_on_cpu(unsigned int cpu, u32 msr_no, u32 l, u32 h);

```

#endif
