Isolation aware smp_call_function/queue_work_on APIs

Marcelo Tosatti (Red Hat)

CPU Isolation MC

Situation with system using CPU isolation

- A number of callpaths which can interrupt isolated CPUs exist, reliance on userspace behaving nicely for interruptions to not occur.

- Guarantee: If distinct interference sources align in time, interference on isolated CPU might sum up (off-topic?).

Proposed changes

- Some of those callpaths are executed from userspace and can therefore return errors.
- Introduce a new cpumask "block_interf_cpumask", with a bit set for the CPUs which should have such interferences blocked.
- Introduce _fail variants of functions that interrupt CPUs, with the variant checking whether CPU is marked as “block inferferences” and returns an error.
- For smp_call_function* family, stop_machine*, queue_work*

Proposed changes pt 2

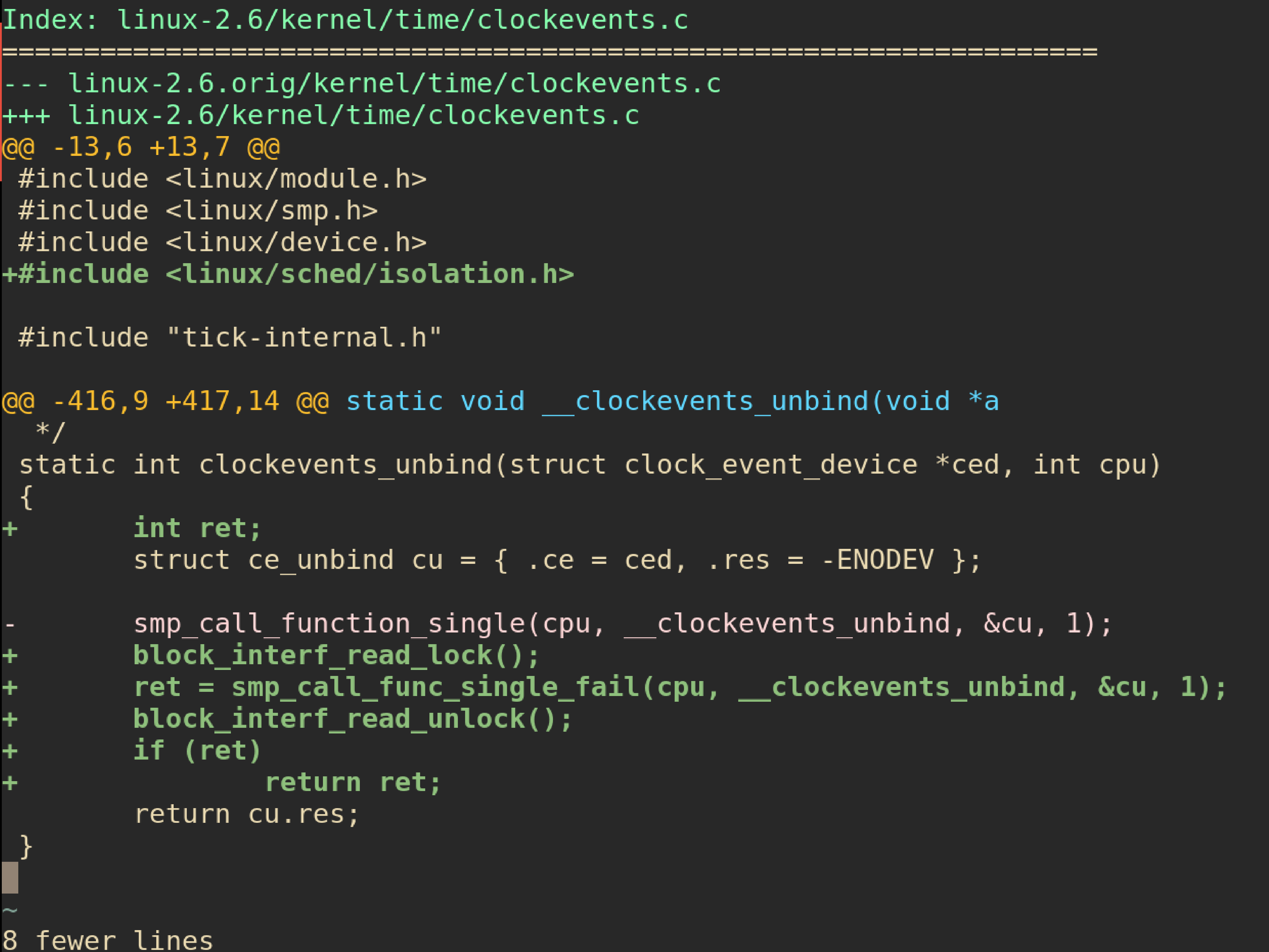
- block_interf_cpumask written from userspace, after system initialization (initialization might require code execution on interference blocked CPUs,for example MTRR initialization, resctrlfs initialization, MSR writes, ...).

Pattern 1

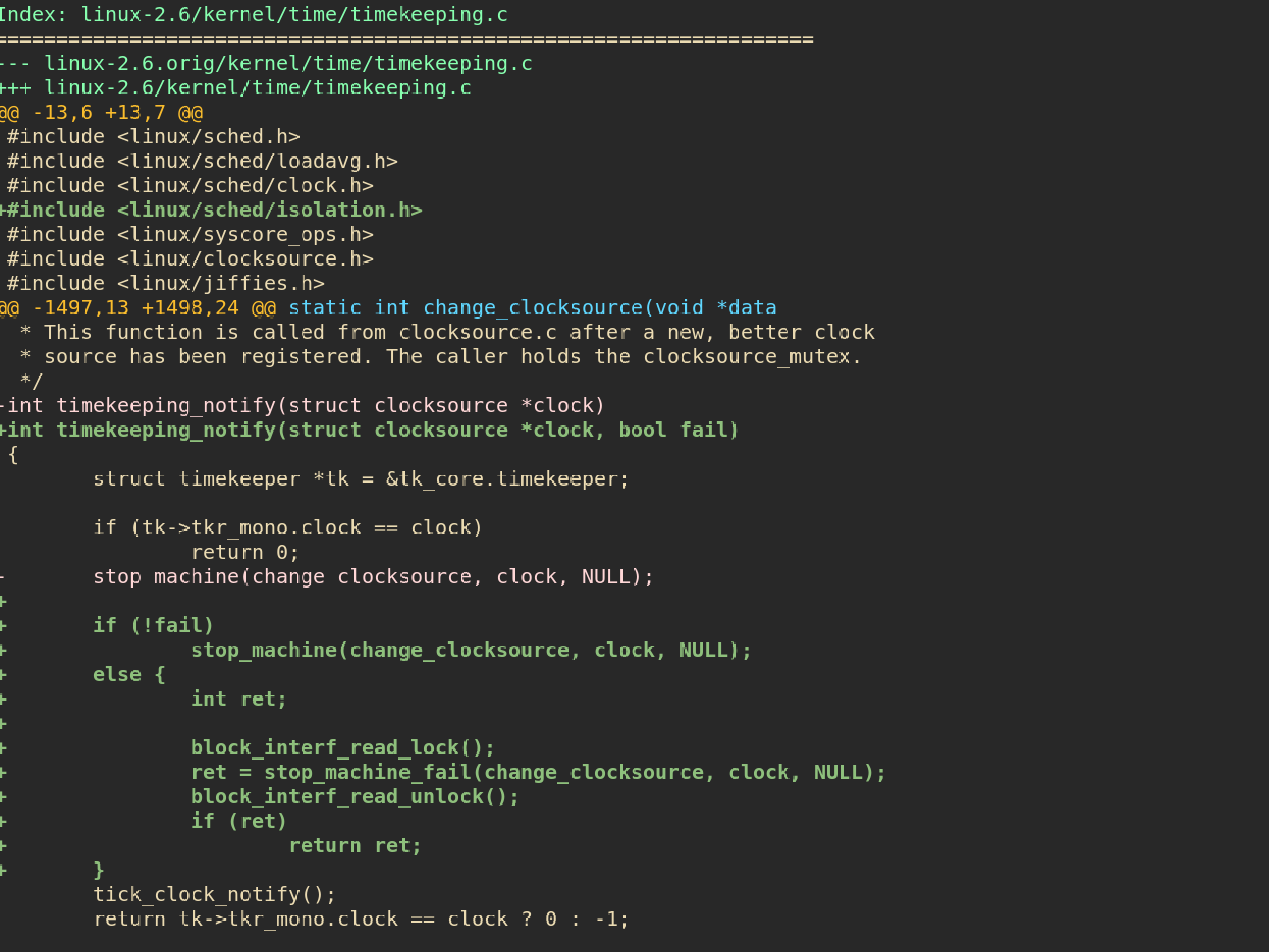
```
block_interf_read_lock(); (per-CPU RWSEM)
...
err = smp_call_func_single_fail();
...
block_interf_read_unlock();

If (ret)
    return err to userspace
```

Pattern 2

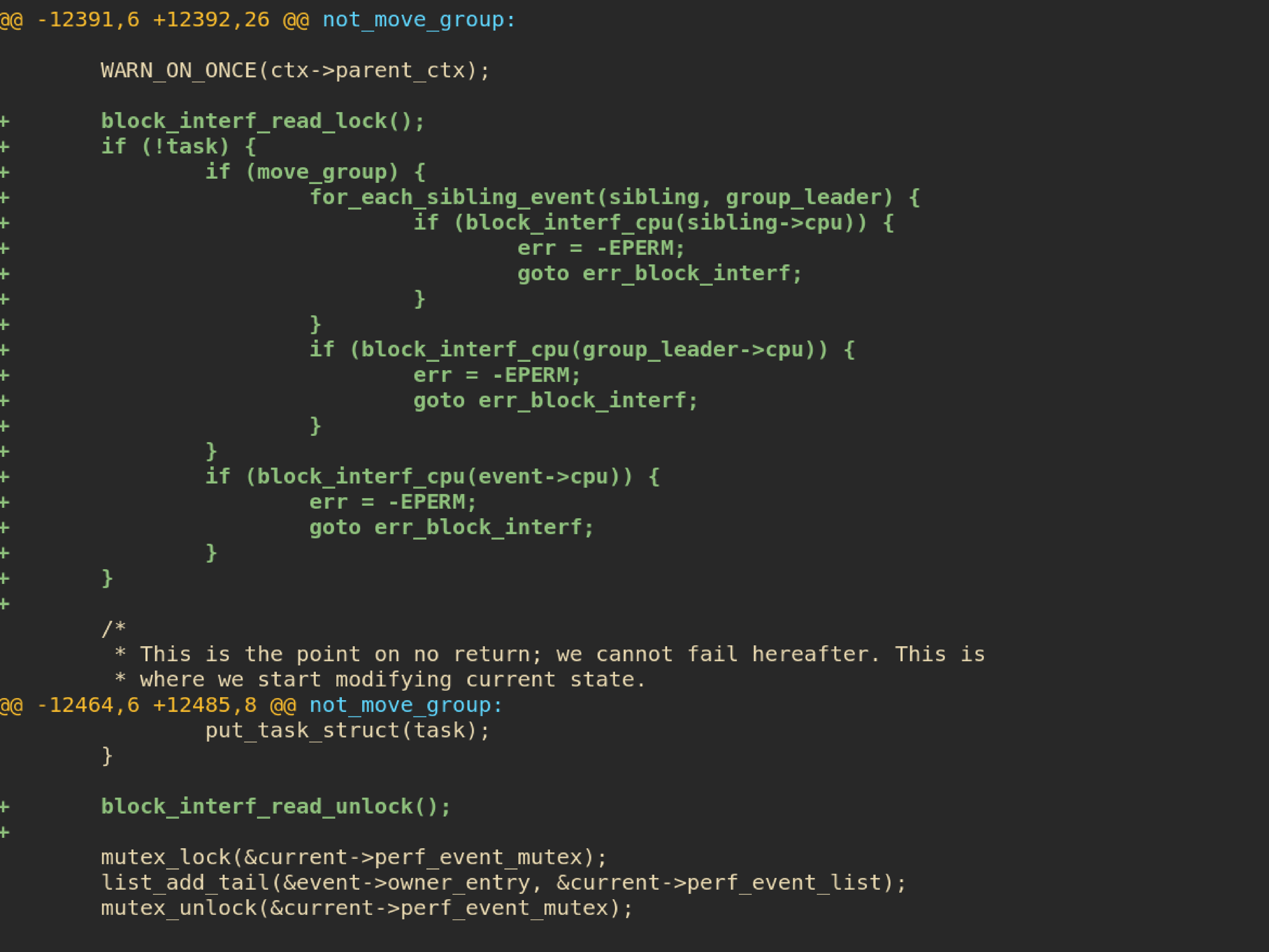
```
block_interf_read_lock(); (per-CPU RWSEM)
...
int cpu = get_target_cpu();
if (cpu_is_blocked_interf(cpu))
    return error to userspace
...
Code to interrupt cpu
...
block_interf_read_unlock();
```



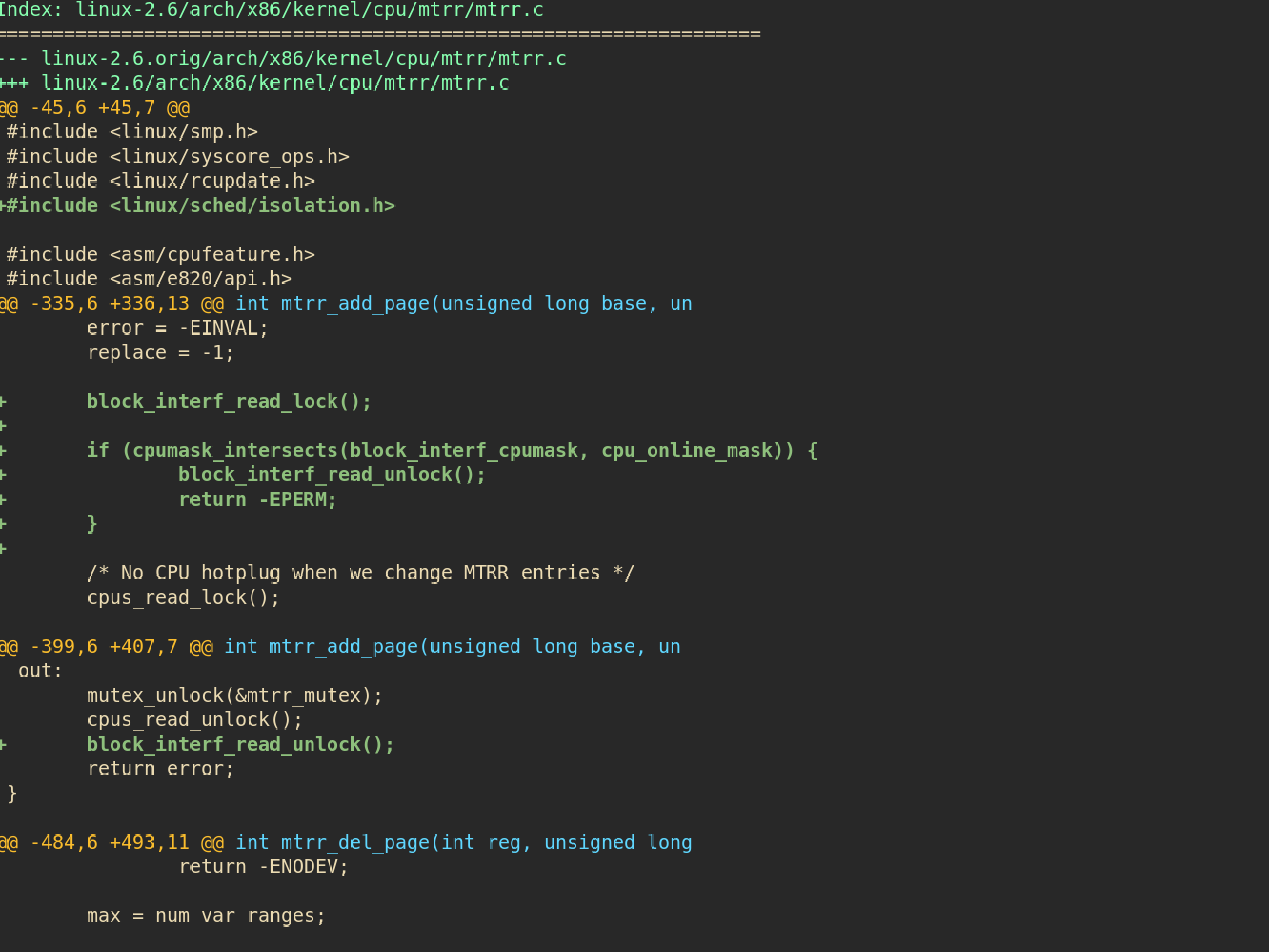
```
Index: linux-2.6/kernel/time/clockevents.c
===================================================================
--- linux-2.6.orig/kernel/time/clockevents.c
+++ linux-2.6/kernel/time/clockevents.c
@@ -13,6 +13,7 @@
 #include <linux/module.h>
 #include <linux/smp.h>
 #include <linux/device.h>
+#include <linux/sched/isolation.h>

 #include "tick-internal.h"

@@ -416,9 +417,14 @@ static void __clockevents_unbind(void *a
  */
 static int clockevents_unbind(struct clock_event_device *ced, int cpu)
 {
+       int ret;
        struct ce_unbind cu = { .ce = ced, .res = -ENODEV };

-       smp_call_function_single(cpu, __clockevents_unbind, &cu, 1);
+       block_interf_read_lock();
+       ret = smp_call_func_single_fail(cpu, __clockevents_unbind, &cu, 1);
+       block_interf_read_unlock();
+       if (ret)
+               return ret;
        return cu.res;
 }

~
8 fewer lines
```



```
Index: linux-2.6/kernel/time/timekeeping.c
===================================================================
--- linux-2.6.orig/kernel/time/timekeeping.c
+++ linux-2.6/kernel/time/timekeeping.c
@@ -13,6 +13,7 @@
 #include <linux/sched.h>
 #include <linux/sched/loadavg.h>
 #include <linux/sched/clock.h>
+#include <linux/sched/isolation.h>
 #include <linux/syscore_ops.h>
 #include <linux/clocksource.h>
 #include <linux/jiffies.h>
@@ -1497,13 +1498,24 @@ static int change_clocksource(void *data
  * This function is called from clocksource.c after a new, better clock
  * source has been registered. The caller holds the clocksource_mutex.
  */
-int timekeeping_notify(struct clocksource *clock)
+int timekeeping_notify(struct clocksource *clock, bool fail)
 {
        struct timekeeper *tk = &tk_core.timekeeper;

        if (tk->tkr_mono.clock == clock)
                return 0;
-       stop_machine(change_clocksource, clock, NULL);
+
+       if (!fail)
+               stop_machine(change_clocksource, clock, NULL);
+       else {
+               int ret;
+
+               block_interf_read_lock();
+               ret = stop_machine_fail(change_clocksource, clock, NULL);
+               block_interf_read_unlock();
+               if (ret)
+                       return ret;
+       }
        tick_clock_notify();
        return tk->tkr_mono.clock == clock ? 0 : -1;
```



```
@@ -12391,6 +12392,26 @@ not_move_group:

        WARN_ON_ONCE(ctx->parent_ctx);

+       block_interf_read_lock();
+       if (!task) {
+               if (move_group) {
+                       for_each_sibling_event(sibling, group_leader) {
+                               if (block_interf_cpu(sibling->cpu)) {
+                                       err = -EPERM;
+                                       goto err_block_interf;
+                               }
+                       }
+                       if (block_interf_cpu(group_leader->cpu)) {
+                               err = -EPERM;
+                               goto err_block_interf;
+                       }
+               }
+               if (block_interf_cpu(event->cpu)) {
+                       err = -EPERM;
+                       goto err_block_interf;
+               }
+       }
+
        /*
         * This is the point on no return; we cannot fail hereafter. This is
         * where we start modifying current state.
@@ -12464,6 +12485,8 @@ not_move_group:
                put_task_struct(task);
        }

+       block_interf_read_unlock();
+
        mutex_lock(&current->perf_event_mutex);
        list_add_tail(&event->owner_entry, &current->perf_event_list);
        mutex_unlock(&current->perf_event_mutex);
```



```
Index: linux-2.6/arch/x86/kernel/cpu/mtrr/mtrr.c
===================================================================
--- linux-2.6.orig/arch/x86/kernel/cpu/mtrr/mtrr.c
+++ linux-2.6/arch/x86/kernel/cpu/mtrr/mtrr.c
@@ -45,6 +45,7 @@
 #include <linux/smp.h>
 #include <linux/syscore_ops.h>
 #include <linux/rcupdate.h>
+#include <linux/sched/isolation.h>

 #include <asm/cpufeature.h>
 #include <asm/e820/api.h>
@@ -335,6 +336,13 @@ int mtrr_add_page(unsigned long base, un
        error = -EINVAL;
        replace = -1;

+       block_interf_read_lock();
+
+       if (cpumask_intersects(block_interf_cpumask, cpu_online_mask)) {
+               block_interf_read_unlock();
+               return -EPERM;
+       }
+
        /* No CPU hotplug when we change MTRR entries */
        cpus_read_lock();

@@ -399,6 +407,7 @@ int mtrr_add_page(unsigned long base, un
  out:
        mutex_unlock(&mtrr_mutex);
        cpus_read_unlock();
+       block_interf_read_unlock();
        return error;
 }

@@ -484,6 +493,11 @@ int mtrr_del_page(int reg, unsigned long
               return -ENODEV;

        max = num_var_ranges;
```

Some discussion topics

- Location for the cpumask to be exposed to userspace?
- Others?