Subject: [PATCH 2/3] powernow-k8: switch to *_on_cpu() functions
Posted by Alexey Dobriyan on Mon, 02 Apr 2007 11:33:27 GMT
View Forum Message <> Reply to Message

Remove changes and restoring of allowed cpu masks as they aren't needed now.

Tested on 2-way Opteron.

Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>
---

 arch/i386/kernel/cpu/cpufreq/powernow-k8.c | 102 ++++++++--------------------
 1 file changed, 29 insertions(+), 73 deletions(-)

--- a/arch/i386/kernel/cpu/cpufreq/powernow-k8.c
+++ b/arch/i386/kernel/cpu/cpufreq/powernow-k8.c
@@ -32,7 +32,6 @@
 #include <linux/slab.h>
 #include <linux/string.h>
 #include <linux/cpumask.h>
-#include <linux/sched.h> /* for current / set_cpus_allowed() */

 #include <asm/msr.h>
 #include <asm/io.h>
@@ -84,17 +83,17 @@ static u32 find_khz_freq_from_fiddid(u32 fid, u32 did)
  return 1000 * find_freq_from_fiddid(fid, did);
 }

-static u32 find_fid_from_pstate(u32 pstate)
+static u32 find_fid_from_pstate(unsigned int cpu, u32 pstate)
 {
  u32 hi, lo;
- rdmsr(MSR_PSTATE_DEF_BASE + pstate, lo, hi);
+ rdmsr_on_cpu(cpu, MSR_PSTATE_DEF_BASE + pstate, &lo, &hi);
  return lo & HW_PSTATE_FID_MASK;
 }

-static u32 find_did_from_pstate(u32 pstate)
+static u32 find_did_from_pstate(unsigned int cpu, u32 pstate)
 {
  u32 hi, lo;
- rdmsr(MSR_PSTATE_DEF_BASE + pstate, lo, hi);
+ rdmsr_on_cpu(cpu, MSR_PSTATE_DEF_BASE + pstate, &lo, &hi);
  return (lo & HW_PSTATE_DID_MASK) >> HW_PSTATE_DID_SHIFT;
 }

@@ -116,14 +115,14 @@ static u32 convert_fid_to_vco_fid(u32 fid)
  * Return 1 if the pending bit is set. Unless we just instructed the processor

```
  * to transition to a new state, seeing this bit set is really bad news.
  */
-static int pending_bit_stuck(void)
+static int pending_bit_stuck(unsigned int cpu)
 {
  u32 lo, hi;

  if (cpu_family == CPU_HW_PSTATE)
   return 0;

- rdmsr(MSR_FIDVID_STATUS, lo, hi);
+ rdmsr_on_cpu(cpu, MSR_FIDVID_STATUS, &lo, &hi);
  return lo & MSR_S_LO_CHANGE_PENDING ? 1 : 0;
 }

@@ -133,13 +132,14 @@ static int pending_bit_stuck(void)
  */
 static int query_current_values_with_pending_wait(struct powernow_k8_data *data)
 {
+ unsigned int cpu = data->cpu;
  u32 lo, hi;
  u32 i = 0;

  if (cpu_family == CPU_HW_PSTATE) {
-  rdmsr(MSR_PSTATE_STATUS, lo, hi);
+  rdmsr_on_cpu(cpu, MSR_PSTATE_STATUS, &lo, &hi);
   i = lo & HW_PSTATE_MASK;
-  rdmsr(MSR_PSTATE_DEF_BASE + i, lo, hi);
+  rdmsr_on_cpu(cpu, MSR_PSTATE_DEF_BASE + i, &lo, &hi);
   data->currfid = lo & HW_PSTATE_FID_MASK;
   data->currdid = (lo & HW_PSTATE_DID_MASK) >> HW_PSTATE_DID_SHIFT;
   return 0;
@@ -149,7 +149,7 @@ static int query_current_values_with_pending_wait(struct
powernow_k8_data *data)
    dprintk("detected change pending stuck\n");
    return 1;
   }
-  rdmsr(MSR_FIDVID_STATUS, lo, hi);
+  rdmsr_on_cpu(cpu, MSR_FIDVID_STATUS, &lo, &hi);
  } while (lo & MSR_S_LO_CHANGE_PENDING);

  data->currvid = hi & MSR_S_HI_CURRENT_VID;
@@ -173,18 +173,18 @@ static void count_off_vst(struct powernow_k8_data *data)
 }

 /* need to init the control msr to a safe value (for each cpu) */
-static void fidvid_msr_init(void)
+static void fidvid_msr_init(unsigned int cpu)
```

```
 {
  u32 lo, hi;
  u8 fid, vid;

- rdmsr(MSR_FIDVID_STATUS, lo, hi);
+ rdmsr_on_cpu(cpu, MSR_FIDVID_STATUS, &lo, &hi);
  vid = hi & MSR_S_HI_CURRENT_VID;
  fid = lo & MSR_S_LO_CURRENT_FID;
  lo = fid | (vid << MSR_C_LO_VID_SHIFT);
  hi = MSR_C_HI_STP_GNT_BENIGN;
  dprintk("cpu%d, init lo 0x%x, hi 0x%x\n", smp_processor_id(), lo, hi);
- wrmsr(MSR_FIDVID_CTL, lo, hi);
+ wrmsr_on_cpu(cpu, MSR_FIDVID_CTL, lo, hi);
 }


@@ -206,7 +206,7 @@ static int write_new_fid(struct powernow_k8_data *data, u32 fid)
   fid, lo, data->plllock * PLL_LOCK_CONVERSION);

  do {
-  wrmsr(MSR_FIDVID_CTL, lo, data->plllock * PLL_LOCK_CONVERSION);
+  wrmsr_on_cpu(data->cpu, MSR_FIDVID_CTL, lo, data->plllock * PLL_LOCK_CONVERSION);
   if (i++ > 100) {
    printk(KERN_ERR PFX "Hardware error - pending bit very stuck - no further pstate changes
possible\n");
    return 1;
@@ -248,7 +248,7 @@ static int write_new_vid(struct powernow_k8_data *data, u32 vid)
   vid, lo, STOP_GRANT_5NS);

  do {
-  wrmsr(MSR_FIDVID_CTL, lo, STOP_GRANT_5NS);
+  wrmsr_on_cpu(data->cpu, MSR_FIDVID_CTL, lo, STOP_GRANT_5NS);
   if (i++ > 100) {
    printk(KERN_ERR PFX "internal error - pending bit very stuck - no further pstate changes
possible\n");
    return 1;
@@ -291,8 +291,8 @@ static int decrease_vid_code_by_step(struct powernow_k8_data *data,
u32 reqvid,
 /* Change hardware pstate by single MSR write */
 static int transition_pstate(struct powernow_k8_data *data, u32 pstate)
 {
- wrmsr(MSR_PSTATE_CTRL, pstate, 0);
- data->currfid = find_fid_from_pstate(pstate);
+ wrmsr_on_cpu(data->cpu, MSR_PSTATE_CTRL, pstate, 0);
+ data->currfid = find_fid_from_pstate(data->cpu, pstate);
  return 0;
 }
```

```
@@ -335,7 +335,7 @@ static int core_voltage_pre_transition(struct powernow_k8_data *data,
u32 reqvid
   smp_processor_id(),
   data->currfid, data->currvid, reqvid, data->rvo);

- rdmsr(MSR_FIDVID_STATUS, lo, maxvid);
+ rdmsr_on_cpu(data->cpu, MSR_FIDVID_STATUS, &lo, &maxvid);
  maxvid = 0x1f & (maxvid >> 16);
  dprintk("ph1 maxvid=0x%x\n", maxvid);
  if (reqvid < maxvid) /* lower numbers are higher voltages */
@@ -499,22 +499,13 @@ static int core_voltage_post_transition(struct powernow_k8_data
*data, u32 reqvi

 static int check_supported_cpu(unsigned int cpu)
 {
- cpumask_t oldmask = CPU_MASK_ALL;
  u32 eax, ebx, ecx, edx;
  unsigned int rc = 0;

- oldmask = current->cpus_allowed;
- set_cpus_allowed(current, cpumask_of_cpu(cpu));
-
- if (smp_processor_id() != cpu) {
-  printk(KERN_ERR PFX "limiting to cpu %u failed\n", cpu);
-  goto out;
- }
-
- if (current_cpu_data.x86_vendor != X86_VENDOR_AMD)
+ if (cpu_data[cpu].x86_vendor != X86_VENDOR_AMD)
   goto out;

- eax = cpuid_eax(CPUID_PROCESSOR_SIGNATURE);
+ eax = cpuid_eax_on_cpu(cpu, CPUID_PROCESSOR_SIGNATURE);
  if (((eax & CPUID_XFAM) != CPUID_XFAM_K8) &&
      ((eax & CPUID_XFAM) < CPUID_XFAM_10H))
   goto out;
@@ -526,20 +517,20 @@ static int check_supported_cpu(unsigned int cpu)
    goto out;
   }

-  eax = cpuid_eax(CPUID_GET_MAX_CAPABILITIES);
+  eax = cpuid_eax_on_cpu(cpu, CPUID_GET_MAX_CAPABILITIES);
   if (eax < CPUID_FREQ_VOLT_CAPABILITIES) {
    printk(KERN_INFO PFX
           "No frequency change capabilities detected\n");
    goto out;
   }
```

```
-  cpuid(CPUID_FREQ_VOLT_CAPABILITIES, &eax, &ebx, &ecx, &edx);
+  cpuid_on_cpu(cpu, CPUID_FREQ_VOLT_CAPABILITIES, &eax, &ebx, &ecx, &edx);
    if ((edx & P_STATE_TRANSITION_CAPABLE) != P_STATE_TRANSITION_CAPABLE) {
     printk(KERN_INFO PFX "Power state transitions not supported\n");
     goto out;
    }
   } else { /* must be a HW Pstate capable processor */
-  cpuid(CPUID_FREQ_VOLT_CAPABILITIES, &eax, &ebx, &ecx, &edx);
+  cpuid_on_cpu(cpu, CPUID_FREQ_VOLT_CAPABILITIES, &eax, &ebx, &ecx, &edx);
    if ((edx & USE_HW_PSTATE) == USE_HW_PSTATE)
     cpu_family = CPU_HW_PSTATE;
    else
@@ -549,7 +540,6 @@ static int check_supported_cpu(unsigned int cpu)
   rc = 1;

  out:
- set_cpus_allowed(current, oldmask);
   return rc;
  }

@@ -849,7 +839,7 @@ static int fill_powernow_table_pstate(struct powernow_k8_data *data,
struct cpuf
     printk(KERN_ERR PFX "invalid pstate %d - bad value %d.\n", i, index);
     printk(KERN_ERR PFX "Please report to BIOS manufacturer\n");
    }
-  rdmsr(MSR_PSTATE_DEF_BASE + index, lo, hi);
+  rdmsr_on_cpu(data->cpu, MSR_PSTATE_DEF_BASE + index, &lo, &hi);
    if (!(hi & HW_PSTATE_VALID_MASK)) {
     dprintk("invalid pstate %d, ignoring\n", index);
     powernow_table[i].frequency = CPUFREQ_ENTRY_INVALID;
@@ -1035,8 +1025,8 @@ static int transition_frequency_pstate(struct powernow_k8_data *data,
unsigned i
   }

   res = transition_pstate(data, pstate);
- data->currfid = find_fid_from_pstate(pstate);
- data->currdid = find_did_from_pstate(pstate);
+ data->currfid = find_fid_from_pstate(data->cpu, pstate);
+ data->currdid = find_did_from_pstate(data->cpu, pstate);
   freqs.new = find_khz_freq_from_fiddid(data->currfid, data->currdid);

   for_each_cpu_mask(i, *(data->available_cores)) {
@@ -1049,7 +1039,6 @@ static int transition_frequency_pstate(struct powernow_k8_data *data,
unsigned i
 /* Driver entry point to switch to the target frequency */
 static int powernowk8_target(struct cpufreq_policy *pol, unsigned targfreq, unsigned relation)
 {
- cpumask_t oldmask = CPU_MASK_ALL;
```

```
  struct powernow_k8_data *data = powernow_data[pol->cpu];
  u32 checkfid;
  u32 checkvid;
@@ -1062,16 +1051,7 @@ static int powernowk8_target(struct cpufreq_policy *pol, unsigned
targfreq, unsi
  checkfid = data->currfid;
  checkvid = data->currvid;

- /* only run on specific CPU from here on */
- oldmask = current->cpus_allowed;
- set_cpus_allowed(current, cpumask_of_cpu(pol->cpu));
-
- if (smp_processor_id() != pol->cpu) {
-  printk(KERN_ERR PFX "limiting to cpu %u failed\n", pol->cpu);
-  goto err_out;
- }
-
- if (pending_bit_stuck()) {
+ if (pending_bit_stuck(pol->cpu)) {
    printk(KERN_ERR PFX "failing targ, change pending bit set\n");
    goto err_out;
   }
@@ -1122,7 +1102,6 @@ static int powernowk8_target(struct cpufreq_policy *pol, unsigned
targfreq, unsi
  ret = 0;

 err_out:
- set_cpus_allowed(current, oldmask);
  return ret;
 }

@@ -1141,7 +1120,6 @@ static int powernowk8_verify(struct cpufreq_policy *pol)
 static int __cpuinit powernowk8_cpu_init(struct cpufreq_policy *pol)
 {
  struct powernow_k8_data *data;
- cpumask_t oldmask = CPU_MASK_ALL;
  int rc;

  if (!cpu_online(pol->cpu))
@@ -1180,16 +1158,7 @@ static int __cpuinit powernowk8_cpu_init(struct cpufreq_policy *pol)
   }
  }

- /* only run on specific CPU from here on */
- oldmask = current->cpus_allowed;
- set_cpus_allowed(current, cpumask_of_cpu(pol->cpu));
-
- if (smp_processor_id() != pol->cpu) {
```

```
-  printk(KERN_ERR PFX "limiting to cpu %u failed\n", pol->cpu);
-  goto err_out;
- }
-
- if (pending_bit_stuck()) {
+ if (pending_bit_stuck(pol->cpu)) {
   printk(KERN_ERR PFX "failing init, change pending bit set\n");
   goto err_out;
  }
@@ -1198,10 +1167,7 @@ static int __cpuinit powernowk8_cpu_init(struct cpufreq_policy *pol)
   goto err_out;

  if (cpu_family == CPU_OPTERON)
-  fidvid_msr_init();
-
- /* run on any CPU again */
- set_cpus_allowed(current, oldmask);
+  fidvid_msr_init(pol->cpu);

  pol->governor = CPUFREQ_DEFAULT_GOVERNOR;
  if (cpu_family == CPU_HW_PSTATE)
@@ -1244,7 +1210,6 @@ static int __cpuinit powernowk8_cpu_init(struct cpufreq_policy *pol)
  return 0;

 err_out:
- set_cpus_allowed(current, oldmask);
  powernow_k8_cpu_exit_acpi(data);

  kfree(data);
@@ -1271,7 +1236,6 @@ static int __devexit powernowk8_cpu_exit (struct cpufreq_policy *pol)
 static unsigned int powernowk8_get (unsigned int cpu)
 {
  struct powernow_k8_data *data;
- cpumask_t oldmask = current->cpus_allowed;
  unsigned int khz = 0;

  data = powernow_data[first_cpu(cpu_core_map[cpu])];
@@ -1279,13 +1243,6 @@ static unsigned int powernowk8_get (unsigned int cpu)
  if (!data)
   return -EINVAL;

- set_cpus_allowed(current, cpumask_of_cpu(cpu));
- if (smp_processor_id() != cpu) {
-  printk(KERN_ERR PFX "limiting to CPU %d failed in powernowk8_get\n", cpu);
-  set_cpus_allowed(current, oldmask);
-  return 0;
- }
-
```

```
  if (query_current_values_with_pending_wait(data))
   goto out;

@@ -1296,7 +1253,6 @@ static unsigned int powernowk8_get (unsigned int cpu)


 out:
- set_cpus_allowed(current, oldmask);
  return khz;
 }
```