

Re: BUG_ON(!cpus_equal(cpumask, tmp));

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-04/0049.html>

From: Andy Whitcroft (*apw_at_shadowen.org*)

Date: 04/01/04

Date: Thu, 01 Apr 2004 01:31:08 +0100

To: "Martin J. Bligh" <mbligh@aracnet.com>, hari@in.ibm.com, Andrew Morton <akpm@osdl.org>

I think we all agree that the reason that we need Hari's change is that the underlying cpu shutdown code is not clearing out all references to the cpu from the VM system; and that in an ideal world we should fix that. But this is v2.6 and we need something which fixes the issue. Hari's patch is a very good starting point, as it closes the window to almost nothing. But there is still a small window in which we can end up hung during shutdown if the timing is wrong as `cpu_online_map` can change whilst we are in the process of sending an IPI. I think we also agree that we don't want to add any additional active locking to VM hot path for the remote tlb invalidate if at all possible.

In a previous email I outlined a possible solution in which we change shutdown to guarantee that if you are in the middle of sending an IPI all cpus online at the start of that IPI will respond, even if they are in the process of going offline.

Below is a patch to 2.6.5-rc3 which implements this idea, subtly modifying Hari's solution and incorporating Martins changes. I have added copious comments as we are essentially open-coding RCU semantics.

I have only tested this on i386 and only shutdown (!) tested it. If there is anyone who can reliably reproduce this issue could try this patch I would appreciate it.

Review/Comments always appreciated.

-apw

```
smp.c | 47 +++++++++++++++++++++++++++++++++++++++++++++++++++++-----
1 files changed, 37 insertions(+), 10 deletions(-)
diff -X /home/apw/lib/vdiff.excl -rupN reference/arch/i386/kernel/smp.c current/arch/i386/kernel/
--- reference/arch/i386/kernel/smp.c      2004-03-11 20:47:12.000000000 +0000
+++ current/arch/i386/kernel/smp.c      2004-04-01 02:01:11.000000000 +0100
@@ -355,8 +355,6 @@ static void flush_tlb_others(cpumask_t c
    */
    BUG_ON(cpus_empty(cpumask));
```

Linux-Kernel: Re: BUG_ON(!cpus_equal(cpumask, tmp));

```
-     cpus_and(tmp, cpumask, cpu_online_map);
-     BUG_ON(!cpus_equal(cpumask, tmp));
-     BUG_ON(cpu_isset(smp_processor_id(), cpumask));
-     BUG_ON(!mm);
@@ -367,16 +365,24 @@ static void flush_tlb_others(cpumask_t c
-     * detected by the NMI watchdog.
-     */
-     spin_lock(&tlbstate_lock);
+
+     /* Subtle, mask the request mask with the currently online cpu's.
+     * Sample this under the lock; cpus in the the middle of going
+     * offline will wait until there is noone in this critical section
+     * before disabling IPI handling. */
+     cpus_and(tmp, cpumask, cpu_online_map);
+     if(cpus_empty(tmp))
+         return;

-     flush_mm = mm;
-     flush_va = va;
- #if NR_CPUS <= BITS_PER_LONG
-     atomic_set_mask(cpumask, &flush_cpumask);
+     atomic_set_mask(tmp, &flush_cpumask);
- #else
-     {
-         int k;
-         unsigned long *flush_mask = (unsigned long *)&flush_cpumask;
-         unsigned long *cpu_mask = (unsigned long *)&cpumask;
+         unsigned long *cpu_mask = (unsigned long *)&tmp;
+         for (k = 0; k < BITS_TO_LONGS(NR_CPUS); ++k)
+             atomic_set_mask(cpu_mask[k], &flush_mask[k]);
-     }
@@ -385,7 +391,7 @@ static void flush_tlb_others(cpumask_t c
-     * We have to send the IPI only to
-     * CPUs affected.
-     */
-     send_IPI_mask(cpumask, INVALIDATE_TLB_VECTOR);
+     send_IPI_mask(tmp, INVALIDATE_TLB_VECTOR);
+     while (!cpus_empty(flush_cpumask))
+         /* nothing. lockup detection does not belong here */
@@ -514,11 +520,8 @@ int smp_call_function (void (*func) (voi
-     */
-     {
-         struct call_data_struct data;
-         int cpus = num_online_cpus()-1;
-
-         if (!cpus)
-             return 0;
-
-         int cpus;
+
+         data.func = func;
+         data.info = info;
+         atomic_set(&data.started, 0);
@@ -527,6 +530,15 @@ int smp_call_function (void (*func) (voi
-         atomic_set(&data.finished, 0);
-         spin_lock(&call_lock);
+
+         /* Subtle, get the current number of online cpus.
+         * Sample this under the lock; cpus in the the middle of going
+         * offline will wait until there is noone in this critical section
+         * before disabling IPI handling. */
+         cpus = num_online_cpus()-1;
```

