

[PATCH] ppc64: hash table races fixes

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-07/6099.html>

From: Benjamin Herrenschmidt (benh_at_kernel.crashing.org)

Date: 07/30/04

To: Andrew Morton <akpm@osdl.org>

Date: Fri, 30 Jul 2004 13:10:17 +1000

Hi !

This patch fixes some possible rare issues with the hash code beeing called with interrupts enabled from `update_mmu_cache`, and fixes some races in the iSeries low level htab code by adding an array of spinlocks

This is actually an old patch already present in SLES kernel and that got "missed" somewhat in the main tree, adapted to recent changes.

Signed-off-by: Benjamin Herrenschmidt <benh@kernel.crashing.org>

```
===== arch/ppc64/kernel/iSeries_htable.c 1.2 vs edited =====
--- 1.2/arch/ppc64/kernel/iSeries_htable.c 2004-02-06 08:10:52 +11:00
+++ edited/arch/ppc64/kernel/iSeries_htable.c 2004-07-29 13:43:52 +10:00
@@ -16,17 +16,26 @@
#include <asm/mmu_context.h>
#include <asm/iSeries/HvCallHpt.h>
#include <asm/abs_addr.h>
-
-#if 0
#include <linux/spinlock.h>
#include <linux/bitops.h>
#include <linux/threads.h>
#include <linux/smp.h>
-
#include <asm/tlbflush.h>
#include <asm/tlb.h>
#include <asm/cputable.h>
#endif
+
+static spinlock_t iSeries_hlocks[64] __cacheline_aligned_in_smp = { [0 ... 63] =
SPIN_LOCK_UNLOCKED };
+
+/*
+ * Very primitive algorithm for picking up a lock
+ */
+static inline void iSeries_hlock(unsigned long slot)
```

Linux–Kernel: [PATCH] ppc64: hash table races fixes

```

+{
+ if (slot & 0x8)
+ slot = ~slot;
+ spin_lock(&iSeries_hlocks[(slot >> 4) & 0x3f]);
+}
+
+static inline void iSeries_hunlock(unsigned long slot)
+{
+ if (slot & 0x8)
+ slot = ~slot;
+ spin_unlock(&iSeries_hlocks[(slot >> 4) & 0x3f]);
+}

static long iSeries_hpte_insert(unsigned long hpte_group, unsigned long va,
                               unsigned long prpn, int secondary,
@@ -44,12 +53,15 @@
    if (secondary)
        return -1;

+ iSeries_hlock(hpte_group);
+
+     slot = HvCallHpt_findValid(&lhpte, va >> PAGE_SHIFT);
- if (lhpte.dw0.dw0.v)
- panic("select_hpte_slot found entry already valid\n");
+ BUG_ON(lhpte.dw0.dw0.v);

- if (slot == -1) /* No available entry found in either group */
+ if (slot == -1) { /* No available entry found in either group */
+ iSeries_hunlock(hpte_group);
+     return -1;
+ }

+     if (slot < 0) { /* MSB set means secondary group */
+         secondary = 1;
@@ -69,6 +81,8 @@
+     /* Now fill in the actual HPTE */
+     HvCallHpt_addValidate(slot, secondary, &lhpte);

+ iSeries_hunlock(hpte_group);
+
+     return (secondary << 3) | (slot & 7);
+ }

@@ -92,6 +106,8 @@
+     /* Pick a random slot to start at */
+     slot_offset = mftb() & 0x7;

+ iSeries_hlock(hpte_group);
+
+     for (i = 0; i < HPTES_PER_GROUP; i++) {
+         lhpte.dw0.dword0 =

```

Linux-Kernel: [PATCH] ppc64: hash table races fixes

```

        iSeries_hpte_getword0(hpte_group + slot_offset);
@@ -99,6 +115,7 @@
        if (!lhpte.dw0.dw0.bolted) {
            HvCallHpt_invalidateSetSwBitsGet(hpte_group +
                slot_offset, 0, 0);
+ iSeries_hunlock(hpte_group);
        return i;
    }

@@ -106,6 +123,8 @@
    slot_offset &= 0x7;
}

+ iSeries_hunlock(hpte_group);
+
    return -1;
}

@@ -121,11 +140,16 @@
    HPTE hpte;
    unsigned long avpn = va >> 23;

+ iSeries_hlock(slot);
+
    HvCallHpt_get(&hpte, slot);
    if ((hpte.dw0.dw0.avpn == avpn) && (hpte.dw0.dw0.v)) {
        HvCallHpt_setPp(slot, (newpp & 0x3) | ((newpp & 0x4) << 1));
+ iSeries_hunlock(slot);
        return 0;
    }
+ iSeries_hunlock(slot);
+
    return -1;
}

@@ -186,11 +210,20 @@
{
    HPTE lhpte;
    unsigned long avpn = va >> 23;
+ unsigned long flags;
+
+ local_irq_save(flags);
+
+ iSeries_hlock(slot);

    lhpte.dw0.dword0 = iSeries_hpte_getword0(slot);

    if ((lhpte.dw0.dw0.avpn == avpn) && lhpte.dw0.dw0.v)
        HvCallHpt_invalidateSetSwBitsGet(slot, 0, 0);
+
+ iSeries_hunlock(slot);

```

Linux-Kernel: [PATCH] ppc64: hash table races fixes

```
+
+ local_irq_restore(flags);
+ }

void hpte_init_iSeries(void)
===== arch/ppc64/kernel/pSeries_htab.c 1.14 vs edited =====
--- 1.14/arch/ppc64/kernel/pSeries_htab.c 2004-04-13 03:54:08 +10:00
+++ edited/arch/ppc64/kernel/pSeries_htab.c 2004-07-29 13:30:16 +10:00
@@ -198,7 +198,6 @@
    HPTTE *hptep = htab_data.htab + slot;
    Hpte_dword0 dw0;
    unsigned long avpn = va >> 23;
- unsigned long flags;
    int ret = 0;

    if (large)
@@ -222,10 +221,10 @@
        tlbier(va);
    } else {
        if (!(cur_cpu_spec->cpu_features & CPU_FTR_LOCKLESS_TLBIE))
- spin_lock_irqsave(&pSeries_tlbier_lock, flags);
+ spin_lock(&pSeries_tlbier_lock);
        tlbier(va, large);
        if (!(cur_cpu_spec->cpu_features & CPU_FTR_LOCKLESS_TLBIE))
- spin_unlock_irqrestore(&pSeries_tlbier_lock, flags);
+ spin_unlock(&pSeries_tlbier_lock);
    }

    return ret;
@@ -275,6 +274,7 @@
    if (large)
        avpn &= ~0x1UL;

+ local_irq_save(flags);
    pSeries_lock_hpte(hptep);

    dw0 = hptep->dw0.dw0;
@@ -292,11 +292,12 @@
        tlbier(va);
    } else {
        if (!(cur_cpu_spec->cpu_features & CPU_FTR_LOCKLESS_TLBIE))
- spin_lock_irqsave(&pSeries_tlbier_lock, flags);
+ spin_lock(&pSeries_tlbier_lock);
        tlbier(va, large);
        if (!(cur_cpu_spec->cpu_features & CPU_FTR_LOCKLESS_TLBIE))
- spin_unlock_irqrestore(&pSeries_tlbier_lock, flags);
+ spin_unlock(&pSeries_tlbier_lock);
    }
+ local_irq_restore(flags);
+ }
```

Linux-Kernel: [PATCH] ppc64: hash table races fixes

```

static void pSeries_flush_hash_range(unsigned long context,
@@ -311,6 +312,8 @@
    /* XXX fix for large ptes */
    unsigned long large = 0;

+ local_irq_save(flags);
+
    j = 0;
    for (i = 0; i < number; i++) {
        if ((batch->addr[i] >= USER_START) &&
@@ -363,7 +366,7 @@
        } else {
            /* XXX double check that it is safe to take this late */
            if (!(cur_cpu_spec->cpu_features & CPU_FTR_LOCKLESS_TLBIE))
- spin_lock_irqsave(&pSeries_tlbie_lock, flags);
+ spin_lock(&pSeries_tlbie_lock);

                asm volatile("ptesync" ::: "memory");

@@ -373,8 +376,10 @@
                asm volatile("eieio; tlbsync; ptesync" ::: "memory");

                if (!(cur_cpu_spec->cpu_features & CPU_FTR_LOCKLESS_TLBIE))
- spin_unlock_irqrestore(&pSeries_tlbie_lock, flags);
+ spin_unlock(&pSeries_tlbie_lock);
        }
    }
+
+ local_irq_restore(flags);
}

void hpte_init_pSeries(void)
===== arch/ppc64/mm/hash_utils.c 1.47 vs edited =====
--- 1.47/arch/ppc64/mm/hash_utils.c 2004-06-10 16:21:41 +10:00
+++ edited/arch/ppc64/mm/hash_utils.c 2004-07-29 13:37:40 +10:00
@@ -251,7 +251,6 @@
    struct mm_struct *mm;
    pte_t *ptep;
    int ret;
- int cpu;
    int user_region = 0;
    int local = 0;
    cpumask_t tmp;
@@ -303,8 +302,7 @@
    if (pgdir == NULL)
        return 1;

- cpu = get_cpu();
- tmp = cpumask_of_cpu(cpu);
+ tmp = cpumask_of_cpu(smp_processor_id());
    if (user_region && cpus_equal(mm->cpu_vm_mask, tmp))
        local = 1;

```

Linux-Kernel: [PATCH] ppc64: hash table races fixes

```
@@ -313,13 +311,10 @@
    ret = hash_huge_page(mm, access, ea, vsid, local);
    else {
        ptep = find_linux_pte(pgdir, ea);
- if (ptep == NULL) {
- put_cpu();
+ if (ptep == NULL)
        return 1;
- }
        ret = __hash_page(ea, access, vsid, ptep, trap, local);
    }
- put_cpu();

    return ret;
}
===== arch/ppc64/mm/init.c 1.73 vs edited =====
--- 1.73/arch/ppc64/mm/init.c 2004-07-25 01:32:48 +10:00
+++ edited/arch/ppc64/mm/init.c 2004-07-29 13:35:21 +10:00
@@ -765,8 +765,8 @@
    void *pgdir;
    pte_t *ptep;
    int local = 0;
- int cpu;
    cpumask_t tmp;
+ unsigned long flags;

    /* handle i-cache coherency */
    if (!(cur_cpu_spec->cpu_features & CPU_FTR_COHERENT_ICACHE) &&
@@ -796,14 +796,14 @@

    vsid = get_vsid(vma->vm_mm->context.id, ea);

- cpu = get_cpu();
- tmp = cpumask_of_cpu(cpu);
+ local_irq_save(flags);
+ tmp = cpumask_of_cpu(smp_processor_id());
    if (cpus_equal(vma->vm_mm->cpu_vm_mask, tmp))
        local = 1;

    __hash_page(ea, pte_val(pte) & (_PAGE_USER|_PAGE_RW), vsid, ptep,
        0x300, local);
- put_cpu();
+ local_irq_restore(flags);
}

void * reserve_phb_iospace(unsigned long size)
```

—
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@vger.kernel.org

Linux-Kernel: [PATCH] ppc64: hash table races fixes

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>