

## [patch] 2.6.6-rc2 Allow architectures to reenale interrupts on contended spinlocks

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-04/5602.html>

---

**From:** Keith Owens (*kaos\_at\_sgi.com*)

**Date:** 04/27/04

To: linux-kernel@vger.kernel.org

Date: Tue, 27 Apr 2004 11:59:51 +1000

Enable interrupts while waiting for a disabled spinlock, but only if interrupts were enabled before issuing `spin_lock_irqsave()`. It makes a measurable difference to interrupt servicing on large systems.

This patch consists of an ia64 specific change (David Mosberger is happy with it) and an architecture independent change. The latter has no effect unless the architecture implements this feature and defines `__HAVE_ARCH_RAW_SPIN_LOCK_FLAGS`. IOW, this change has no effect on anything except ia64, unless the other architecture maintainers want to implement this feature for their architecture.

Index: 2.6.6-rc2/arch/ia64/kernel/head.S

-----  
--- 2.6.6-rc2.orig/arch/ia64/kernel/head.S Mon Apr 5 11:02:53 2004

+++ 2.6.6-rc2/arch/ia64/kernel/head.S Tue Apr 27 11:52:00 2004

@@ -866,12 +866,14 @@

\* Inputs:

\* ar.pfs - saved CFM of caller

\* ar.ccv - 0 (and available for use)

+ \* r27 - flags from `spin_lock_irqsave` or 0. Must be preserved.

\* r28 - available for use.

\* r29 - available for use.

\* r30 - available for use.

\* r31 - address of lock, available for use.

\* b6 - return address

\* p14 - available for use.

+ \* p15 - used to track flag status.

\*

\* If you patch this code to use more registers, do not forget to update

\* the clobber lists for `spin_lock()` in `include/asm-ia64/spinlock.h`.

@@ -885,22 +887,27 @@

.save rp, r28

.body

nop 0

- nop 0

## Linux–Kernel: [patch] 2.6.6–rc2 Allow architectures to reenale interrupts on contended spinlocks

```
+ tbit.nz p15,p0=r27,IA64_PSR_I_BIT
    .restore sp // pop existing prologue after next insn
    mov b6 = r28
    .prologue
    .save ar.pfs, r0
    .altrp b6
    .body
+ ;;
+(p15) ssm psr.i // reenale interrupts if they were on
+ ;;
+(p15) srlz.d
    .wait:
        // exponential backoff, kdb, lockmeter etc. go in here
        hint @pause
        ld4 r30=[r31] // don't use ld4.bias; if it's contended, we won't write the word
        nop 0
        ;;
- cmp4.eq p14,p0=r30,r0
-(p14) br.cond.sptk.few b6 // lock is now free, try to acquire
- br.cond.sptk.few .wait
+ cmp4.ne p14,p0=r30,r0
+(p14) br.cond.sptk.few .wait
+(p15) rsm psr.i // disable interrupts if we reenaled them
+ br.cond.sptk.few b6 // lock is now free, try to acquire
END(ia64_spinlock_contention_pre3_4)

#else
@@ -909,14 +916,21 @@
    .prologue
    .altrp b6
    .body
+ tbit.nz p15,p0=r27,IA64_PSR_I_BIT
+ ;;
    .wait:
+(p15) ssm psr.i // reenale interrupts if they were on
+ ;;
+(p15) srlz.d
+.wait2:
    // exponential backoff, kdb, lockmeter etc. go in here
    hint @pause
    ld4 r30=[r31] // don't use ld4.bias; if it's contended, we won't write the word
    ;;
    cmp4.ne p14,p0=r30,r0
    mov r30 = 1
-(p14) br.cond.sptk.few .wait
+(p14) br.cond.sptk.few .wait2
+(p15) rsm psr.i // disable interrupts if we reenaled them
    ;;
    cmpxchg4.acq r30=[r31], r30, ar.ccv
    ;;
Index: 2.6.6–rc2/include/asm–ia64/spinlock.h
```

```

-----
--- 2.6.6-rc2.orig/include/asm-ia64/spinlock.h Mon Apr 5 11:04:32 2004
+++ 2.6.6-rc2/include/asm-ia64/spinlock.h Tue Apr 27 11:48:03 2004
@@ -32,10 +32,12 @@
 * carefully coded to touch only those registers that spin_lock() marks "clobbered".
 */

-#define IA64_SPINLOCK_CLOBBERS "ar.ccv", "ar.pfs", "p14", "r28", "r29", "r30", "b6", "memory"
+#define IA64_SPINLOCK_CLOBBERS "ar.ccv", "ar.pfs", "p14", "p15", "r27", "r28", "r29", "r30", "b6",
"memory"
+
+#define __HAVE_ARCH_RAW_SPIN_LOCK_FLAGS

static inline void
-_raw_spin_lock (spinlock_t *lock)
+_raw_spin_lock_flags (spinlock_t *lock, unsigned long flags)
{
    register volatile unsigned int *ptr asm ("r31") = &lock->lock;

@@ -50,9 +52,10 @@
    "cmpxchg4.acq r30 = [%1], r30, ar.ccv\n\t"
    "movl r29 = ia64_spinlock_contention_pre3_4;;\n\t"
    "cmp4.ne p14, p0 = r30, r0\n\t"
- "mov b6 = r29;;\n"
+ "mov b6 = r29;;\n\t"
+ "mov r27=%2\n\t"
    "(p14) br.cond.spnt.many b6"
- : "=r"(ptr) : "r"(ptr) : IA64_SPINLOCK_CLOBBERS);
+ : "=r"(ptr) : "r"(ptr), "r" (flags) : IA64_SPINLOCK_CLOBBERS);
# else
    asm volatile ("{\n\t"
        " mov ar.ccv = r0\n\t"
@@ -60,29 +63,32 @@
        " mov r30 = 1;;\n\t"
        "}\n\t"
        "cmpxchg4.acq r30 = [%1], r30, ar.ccv;;\n\t"
- "cmp4.ne p14, p0 = r30, r0\n"
+ "cmp4.ne p14, p0 = r30, r0\n\t"
+ "mov r27=%2\n\t"
        "(p14) brl.cond.spnt.many ia64_spinlock_contention_pre3_4;;"
- : "=r"(ptr) : "r"(ptr) : IA64_SPINLOCK_CLOBBERS);
+ : "=r"(ptr) : "r"(ptr), "r" (flags) : IA64_SPINLOCK_CLOBBERS);
# endif /* CONFIG_MCKINLEY */
# else
# ifdef CONFIG_ITANIUM
    /* don't use brl on Itanium... */
    /* mis-declare, so we get the entry-point, not it's function descriptor: */
    asm volatile ("mov r30 = 1\n\t"
+ "mov r27=%2\n\t"
        "mov ar.ccv = r0;;\n\t"
        "cmpxchg4.acq r30 = [%0], r30, ar.ccv\n\t"

```

Linux-Kernel: [patch] 2.6.6-rc2 Allow architectures to reenale interrupts on contended spinlocks

```
        "movl r29 = ia64_spinlock_contention;;\n\t"
        "cmp4.ne p14, p0 = r30, r0\n\t"
- "mov b6 = r29;;\n"
+ "mov b6 = r29;;\n\t"
        "(p14) br.call.spnt.many b6 = b6"
- : "=r"(ptr) : "r"(ptr) : IA64_SPINLOCK_CLOBBERS);
+ : "=r"(ptr) : "r"(ptr), "r" (flags) : IA64_SPINLOCK_CLOBBERS);
# else
    asm volatile ("mov r30 = 1\n\t"
+ "mov r27=%2\n\t"
        "mov ar.ccv = r0;;\n\t"
        "cmpxchg4.acq r30 = [%0], r30, ar.ccv;;\n\t"
        "cmp4.ne p14, p0 = r30, r0\n\t"
        "(p14) brl.call.spnt.many b6=ia64_spinlock_contention;;"
- : "=r"(ptr) : "r"(ptr) : IA64_SPINLOCK_CLOBBERS);
+ : "=r"(ptr) : "r"(ptr), "r" (flags) : IA64_SPINLOCK_CLOBBERS);
# endif /* CONFIG_MCKINLEY */
#endif
}
```

Index: 2.6.6-rc2/include/linux/spinlock.h

```
=====
--- 2.6.6-rc2.orig/include/linux/spinlock.h Thu Dec 18 13:58:49 2003
+++ 2.6.6-rc2/include/linux/spinlock.h Tue Apr 27 11:48:03 2004
@@ -184,6 +184,12 @@

#endif /* !SMP */

+#ifdef __HAVE_ARCH_RAW_SPIN_LOCK_FLAGS
+#define _raw_spin_lock(lock) _raw_spin_lock_flags(lock, 0)
+#else
+#define _raw_spin_lock_flags(lock, flags) do { (void)flags; _raw_spin_lock(lock); } while(0)
+#endif
+
+/*
+ * Define the various spin_lock and rw_lock methods. Note we define these
+ * regardless of whether CONFIG_SMP or CONFIG_PREEMPT are set. The various
@@ -257,7 +263,7 @@
do { \
    local_irq_save(flags); \
    preempt_disable(); \
- _raw_spin_lock(lock); \
+ _raw_spin_lock_flags(lock, flags); \
} while (0)
```

```
#define spin_lock_irq(lock) \
```

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org  
More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Linux–Kernel: [patch] 2.6.6–rc2 Allow architectures to reenale interrupts on contended spinlocks

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