

[PATCH] Microstate accounting for 2.6.0-test1

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2003-07/0398.html>

From: Peter Chubb (peterc_at_gelato.unsw.edu.au)

Date: 07/22/03

To: linux-kernel@vger.kernel.org

Date: Tue, 22 Jul 2003 13:27:50 +1000

Hi Folks,

Here's the latest and greatest Microstate accounting patch, against the 2.6.0-test1 kernel as of this morning my time. Tested on IA64 and I386.

When applied, and microstate accounting is enabled, the patch adds a new file `/proc/pid/msa` that contains the time in cycles for each state. As a bonus, you get the time spent in each interrupt handler.

Sample output on a 2.5G P4:

```
; cat /proc/$$/msa
State: Interruptible
ONCPU_USER 65290612
ONCPU_SYS 32495176
INTERRUPTIBLE 41788648905302
UNINTERRUPTIBLE 885657552
INTERRUPTED 2225408
ACTIVEQUEUE 1539599360
EXPIREDQUEUE 0
STOPPED 0
ZOMBIE 0
SLP_POLL 0
SLP_PAGING 0
SLP_FUTEX 0

$ cat /proc/interrupts
CPU0
0: 77361890 648007534 IO-APIC-edge timer
2: 0 0 XT-PIC cascade
3: 0 0 IO-APIC-edge serial
4: 2301 60917 IO-APIC-edge serial
8: 4 25 IO-APIC-edge rtc
9: 14 188 IO-APIC-level acpi
14: 291288 4067531 IO-APIC-edge ide0
15: 18 158 IO-APIC-edge ide1
```

Linux-Kernel: [PATCH] Microstate accounting for 2.6.0-test1

```
16: 178246 1652739 IO-APIC-level uhci-hcd
18: 73399 579808 IO-APIC-level uhci-hcd, eth0
19: 31 139 IO-APIC-level uhci-hcd
23: 0 0 IO-APIC-level ehci_hcd
NMI: 0
LOC: 77367218
ERR: 0
MIS: 0
```

(the third column is time in nanoseconds)

The patch also adds a new system call, which gives scaled (nanosecond) numbers.

```
msa(n, select, timers)
int n;
enum { MSA_SELF, MSA_CHILDREN } select,
clk_t timers[n]
```

fills the n timers with the number of nanoseconds in that state, for the current process, or for waited-for children.

GOTCHAS:

— I've measured the overhead as around 5% per context switch, but negligible for real workloads.

— Time spent in the kernel as a result of a trap (e.g., when paging) is accounted for as user time (which means that time reported as ONCPU_SYS is *only* system call time)

— If you're on a machine where the CPU frequency changes, the results are useless. You really need Dominik Brodowski's patch from <http://marc.theaimsgroup.com/?l=linux-kernel&m=105860269801212&q=raw> with corresponding changes to `asm-i386/msa.h`

```
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5--import/arch/i386/Kconfig linux-2.5--ustate/arch/i386/Kconfig
--- linux-2.5--import/arch/i386/Kconfig Mon Jul 14 11:12:14 2003
+++ linux-2.5--ustate/arch/i386/Kconfig Mon Jul 14 15:15:29 2003
@@ -1383,6 +1383,15 @@
     depends on X86_LOCAL_APIC && !X86_VISWS
     default y
```

```
+config MICROSTATE
+bool "Microstate accounting"
+help
+ This option causes the kernel to keep very accurate track of
+ how long your threads spend on the runqueues, running, or asleep or
+ stopped. It will slow down your kernel.
+ Times are reported in /proc/pid/msa and through a new msa()
+ system call.
```

```

+
+endmenu

source "security/Kconfig"
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/arch/i386/kernel/entry.S linux-2.5-ustate/arch/i386/kernel/entry.S
--- linux-2.5-import/arch/i386/kernel/entry.S Mon Jul 14 11:12:14 2003
+++ linux-2.5-ustate/arch/i386/kernel/entry.S Mon Jul 14 15:15:29 2003
@@ -264,9 +264,17 @@

        testb $_TIF_SYSCALL_TRACE, TI_FLAGS(%ebp)
        jnz syscall_trace_entry
+#ifdef CONFIG_MICROSTATE
+ pushl %eax
+ call msa_start_syscall
+ popl %eax
+#endif
        call *sys_call_table(,%eax,4)
        movl %eax, EAX(%esp)
        cli
+#ifdef CONFIG_MICROSTATE
+ call msa_end_syscall
+#endif
        movl TI_FLAGS(%ebp), %ecx
        testw $_TIF_ALLWORK_MASK, %cx
        jne syscall_exit_work
@@ -288,9 +296,17 @@
        testb $_TIF_SYSCALL_TRACE, TI_FLAGS(%ebp)
        jnz syscall_trace_entry
syscall_call:
+#ifdef CONFIG_MICROSTATE
+ pushl %eax
+ call msa_start_syscall
+ popl %eax
+#endif
        call *sys_call_table(,%eax,4)
        movl %eax, EAX(%esp) # store the return value
syscall_exit:
+#ifdef CONFIG_MICROSTATE
+ call msa_end_syscall
+#endif
        cli # make sure we don't miss an interrupt
            # setting need_resched or sigpending
            # between sampling and the iret
@@ -878,5 +894,6 @@
        .long sys_fstatfs64
        .long sys_tgkill /* 270 */
        .long sys_utimes
+ .long sys_msa /* 272 */

nr_syscalls=(.-sys_call_table)/4

```

Linux-Kernel: [PATCH] Microstate accounting for 2.6.0-test1

```
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/arch/i386/kernel/irq.c linux-2.5-ustate/arch/i386/kernel/irq.c
--- linux-2.5-import/arch/i386/kernel/irq.c Tue Jul 8 08:32:12 2003
+++ linux-2.5-ustate/arch/i386/kernel/irq.c Tue Jul 8 09:19:05 2003
@@ -157,10 +157,18 @@
     seq_printf(p, "%3d: ",i);
 #ifndef CONFIG_SMP
     seq_printf(p, "%10u ", kstat_irqs(i));
+#ifdef CONFIG_MICROSTATE
+ seq_printf(p, "%10llu", msa_irq_time(0, i));
+#endif
 #else
     for (j = 0; j < NR_CPUS; j++)
- if (cpu_online(j))
+ if (cpu_online(j)) {
     seq_printf(p, "%10u ", kstat_cpu(j).irqs[i]);
+#ifdef CONFIG_MICROSTATE
+ seq_printf(p, "%10llu", msa_irq_time(j, i));
+#endif
+ }
+
 #endif
     seq_printf(p, " %14s", irq_desc[i].handler->typename);
     seq_printf(p, " %s", action->name);
@@ -421,6 +429,7 @@
     unsigned int status;

     irq_enter();
+ msa_start_irq(irq);

 #ifdef CONFIG_DEBUG_STACKOVERFLOW
     /* Debugging check for stack overflow: is there less than 1KB free? */
@@ -500,6 +509,7 @@
     spin_unlock(&desc->lock);

     irq_exit();
+ msa_finish_irq(irq);

     return 1;
 }
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/arch/ia64/Kconfig linux-2.5-ustate/arch/ia64/Kconfig
--- linux-2.5-import/arch/ia64/Kconfig Mon Jul 14 11:12:15 2003
+++ linux-2.5-ustate/arch/ia64/Kconfig Mon Jul 14 15:15:29 2003
@@ -740,6 +740,15 @@
     and restore instructions. It's useful for tracking down spinlock
     problems, but slow! If you're unsure, select N.

+config MICROSTATE
+ bool "Microstate accounting"
+ help
```

Linux-Kernel: [PATCH] Microstate accounting for 2.6.0-test1

- + This option causes the kernel to keep very accurate track of
- + how long your threads spend on the runqueues, running, or asleep or
- + stopped. It will slow down your kernel.
- + Times are reported in /proc/pid/msa and through a new msa()
- + system call.
- +
- + endmenu

```
source "security/Kconfig"
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/arch/ia64/kernel/entry.S linux-2.5-ustate/arch/ia64/kernel/entry.S
---- linux-2.5-import/arch/ia64/kernel/entry.S Fri Jul 18 08:27:52 2003
+++ linux-2.5-ustate/arch/ia64/kernel/entry.S Tue Jul 22 09:28:54 2003
@@ -539,6 +539,36 @@
     br.cond.sptk strace_save_retval
END(ia64_trace_syscall)
```

```
+#ifdef CONFIG_MICROSTATE
+GLOBAL_ENTRY(invoke_msa_end_syscall)
+.prologue ASM_UNW_PRLG_RP|ASM_UNW_PRLG_PFS, ASM_UNW_PRLG_GRSAVE(8)
+.alloc loc1=ar.pfs,8,4,0,0
+.mov loc0=rp
+.body
+;;
+.br.call.sptk.many rp=msa_end_syscall
+1: .mov rp=loc0
+.mov ar.pfs=loc1
+.br.ret.sptk.many rp
+END(invoke_msa_end_syscall)
+
+GLOBAL_ENTRY(invoke_msa_start_syscall)
+.prologue ASM_UNW_PRLG_RP|ASM_UNW_PRLG_PFS, ASM_UNW_PRLG_GRSAVE(8)
+.alloc loc1=ar.pfs,8,4,0,0
+.mov loc0=rp
+.body
+.mov loc2=b6
+.mov loc3=r15
+;;
+.br.call.sptk.many rp=msa_start_syscall
+1: .mov rp=loc0
+.mov r15=loc3
+.mov ar.pfs=loc1
+.mov b6=loc2
+.br.ret.sptk.many rp
+END(invoke_msa_start_syscall)
+#endif /* CONFIG_MICROSTATE */
+
+GLOBAL_ENTRY(ia64_ret_from_clone)
+    PT_REGS_UNWIND_INFO(0)
+    { /*
@@ -620,6 +650,10 @@
```

```

*/
GLOBAL_ENTRY(ia64_leave_syscall)
    PT_REGS_UNWIND_INFO(0)
+#ifdef CONFIG_MICROSTATE
+ br.call.sptk.many rp=invoke_msa_end_syscall
+1:
+#endif
    /*
     * work.need_resched etc. mustn't get changed by this CPU before it returns to
     * user- or fsys-mode, hence we disable interrupts early on:
@@ -961,7 +995,7 @@
    mov loc7=0
    (pRecurse) br.call.sptk.few b0=rse_clear_invalid
    ;;
- mov loc8=0
+1: mov loc8=0
    mov loc9=0
    cmp.ne pReturn,p0=r0,in1 // if recursion count != 0, we need to do a br.ret
    mov loc10=0
@@ -1461,7 +1495,7 @@
    data8 sys_clock_nanosleep
    data8 sys_fstats64
    data8 sys_stats64
- data8 ia64_ni_syscall
+ data8 sys_msa
    data8 ia64_ni_syscall // 1260
    data8 ia64_ni_syscall
    data8 ia64_ni_syscall
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/arch/ia64/kernel/irq_ia64.c linux-2.5-ustate/arch/ia64/kernel/irq_ia64.c
--- linux-2.5-import/arch/ia64/kernel/irq_ia64.c Fri Jun 27 09:17:14 2003
+++ linux-2.5-ustate/arch/ia64/kernel/irq_ia64.c Tue Jul 22 09:28:56 2003
@@ -76,6 +76,7 @@
ia64_handle_irq (ia64_vector vector, struct pt_regs *regs)
{
    unsigned long saved_tpr;
+ ia64_vector oldvector;
#ifdef CONFIG_SMP
# define IS_RESCCHEDULE(vec) (vec == IA64_IPI_RESCCHEDULE)
#else
@@ -119,6 +120,8 @@
    */
    saved_tpr = ia64_get_tpr();
    ia64_srlz_d();
+ oldvector = vector;
+ msa_start_irq(local_vector_to_irq(vector));
    while (vector != IA64_SPURIOUS_INT_VECTOR) {
        if (!IS_RESCCHEDULE(vector)) {
            ia64_set_tpr(vector);
@@ -133,7 +136,10 @@
            ia64_set_tpr(saved_tpr);

```

Linux-Kernel: [PATCH] Microstate accounting for 2.6.0-test1

```

    }
    ia64_eoi();
- vector = ia64_get_ivr();
+ oldvector = vector;
+ vector = ia64_get_ivr();
+ msa_continue_irq(local_vector_to_irq(oldvector),
+ local_vector_to_irq(vector));
    }
    /*
     * This must be done *after* the ia64_eoi(). For example, the keyboard softirq
@@ -142,6 +148,8 @@
    */
    if (local_softirq_pending())
        do_softirq();
+
+ msa_finish_irq(local_vector_to_irq(vector));
}

#ifdef CONFIG_SMP
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/arch/ia64/kernel/ivt.S linux-2.5-ustate/arch/ia64/kernel/ivt.S
--- linux-2.5-import/arch/ia64/kernel/ivt.S Tue Jun 24 15:12:05 2003
+++ linux-2.5-ustate/arch/ia64/kernel/ivt.S Tue Jul 22 09:28:56 2003
@@ -697,6 +697,10 @@
    srlz.i // guarantee that interruption collection is on
    ;;
    (p15) ssm psr.i // restore psr.i
+#ifdef CONFIG_MICROSTATE
+ br.call.sptk.many rp=invoke_msa_start_syscall
+1:
+#endif /* CONFIG_MICROSTATE */
    ;;
    mov r3=NR_syscalls - 1
    movl r16=sys_call_table
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/fs/proc/base.c linux-2.5-ustate/fs/proc/base.c
--- linux-2.5-import/fs/proc/base.c Mon Jul 14 11:12:20 2003
+++ linux-2.5-ustate/fs/proc/base.c Tue Jul 22 09:28:57 2003
@@ -65,6 +65,9 @@
    PROC_PID_ATTR_EXEC,
    PROC_PID_ATTR_FSCREATE,
#endif
+#ifdef CONFIG_MICROSTATE
+ PROC_PID_MSA,
+#endif
    PROC_PID_FD_DIR = 0x8000, /* 0x8000-0xffff */
};

@@ -95,6 +98,9 @@
#ifdef CONFIG_KALLSYMS
    E(PROC_PID_WCHAN, "wchan", S_IFREG|S_IRUGO),

```

```

#endif
+#ifdef CONFIG_MICROSTATE
+ E(PROC_PID_MSA, "msa", S_IFREG|S_IRUGO),
+#endif
    {0,0,NULL,0}
};
#ifdef CONFIG_SECURITY
@@ -288,6 +294,60 @@
}
#endif /* CONFIG_KALLSYMS */

+#ifdef CONFIG_MICROSTATE
+/*
+ * provides microstate accounting information
+ *
+ */
+static int proc_pid_msa(struct task_struct *task, char *buffer)
+{
+ struct microstates *msp = &task->microstates;
+ static char *statenames[] = {
+ "User",
+ "System",
+ "Interruptible",
+ "Uninterruptible",
+ "OnActiveQueue",
+ "OnExpiredQueue",
+ "Zombie",
+ "Stopped",
+ "Paging",
+ "Futex",
+ "Poll",
+ "Interrupted",
+ };
+
+ return sprintf(buffer,
+ "State: %s\n" \
+ "ONCPU_USER %15llu\n" \
+ "ONCPU_SYS %15llu\n" \
+ "INTERRUPTIBLE %15llu\n" \
+ "UNINTERRUPTIBLE%15llu\n" \
+ "INTERRUPTED %15llu\n" \
+ "ACTIVEQUEUE %15llu\n" \
+ "EXPIREDQUEUE %15llu\n" \
+ "STOPPED %15llu\n" \
+ "ZOMBIE %15llu\n" \
+ "SLP_POLL %15llu\n" \
+ "SLP_PAGING %15llu\n" \
+ "SLP_FUTEX %15llu\n", \
+ msp->cur_state >= 0 && msp->cur_state < NR_MICRO_STATES ?
+ statenames[msp->cur_state] : "Impossible",
+ (unsigned long long)msp->timers[ONCPU_USER],

```


Linux-Kernel: [PATCH] Microstate accounting for 2.6.0-test1

```

linux-2.5-import/include/asm-i386/msa.h linux-2.5-ustate/include/asm-i386/msa.h
--- linux-2.5-import/include/asm-i386/msa.h Thu Jan 1 10:00:00 1970
+++ linux-2.5-ustate/include/asm-i386/msa.h Tue Jul 22 12:58:57 2003
@@ -0,0 +1,29 @@
+/******
+ * asm-i386/msa.h
+ *
+ * Provide an architecture-specific clock.
+ */
+
+#ifndef _ASM_I386_MSA_H
+# define _ASM_I386_MSA_H
+
+# ifdef __KERNEL__
+# include <linux/config.h>
+
+# if defined(CONFIG_X86_TSC)
+# include <asm/msr.h>
+# include <asm/div64.h>
+# define MSA_NOW(now) rdtscll(now)
+
+extern unsigned long cpu_khz;
+# define MSA_TO_NSEC(clk) ({ clk_t_x = ((clk) * 1000000ULL); do_div(_x, cpu_khz); _x; })
+
+# else
+unsigned long long monotonic_clock(void);
+# define MSA_NOW(now) do { now = monotonic_clock(); } while (0)
+# define MSA_TO_NSEC(clk) (clk)
+# endif
+
+# endif /* _KERNEL */
+
+#endif /* _ASM_I386_MSA_H */
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/include/asm-i386/unistd.h linux-2.5-ustate/include/asm-i386/unistd.h
--- linux-2.5-import/include/asm-i386/unistd.h Mon Jul 14 11:12:20 2003
+++ linux-2.5-ustate/include/asm-i386/unistd.h Mon Jul 14 15:15:41 2003
@@ -277,6 +277,7 @@
#define __NR_fstatfs64 269
#define __NR_tgkill 270
#define __NR_utimes 271
+#define __NR_msa 272

#define NR_syscalls 272

diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/include/asm-ia64/msa.h linux-2.5-ustate/include/asm-ia64/msa.h
--- linux-2.5-import/include/asm-ia64/msa.h Thu Jan 1 10:00:00 1970
+++ linux-2.5-ustate/include/asm-ia64/msa.h Wed Jul 2 09:49:23 2003
@@ -0,0 +1,21 @@
+/******

```

Linux-Kernel: [PATCH] Microstate accounting for 2.6.0-test1

```

+ * asm-ia64/msa.h
+ *
+ * Provide an architecture-specific clock.
+ */
+
+#ifndef _ASM_IA64_MSA_H
+#define _ASM_IA64_MSA_H
+
+#ifdef __KERNEL__
+#include <asm/processor.h>
+#include <asm/timex.h>
+#include <asm/smp.h>
+
+#define MSA_NOW(now) do { now = (clk_t)get_cycles(); } while (0)
+
+#define MSA_TO_NSEC(clk) ((1000000000*clk) / cpu_data(smp_processor_id())->itc_freq)
+
+#endif /* __KERNEL */
+
+#endif /* _ASM_IA64_MSA_H */
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/include/asm-ia64/unistd.h linux-2.5-ustate/include/asm-ia64/unistd.h
--- linux-2.5-import/include/asm-ia64/unistd.h Fri Jul 11 10:10:32 2003
+++ linux-2.5-ustate/include/asm-ia64/unistd.h Mon Jul 14 15:15:41 2003
@@ -248,10 +248,11 @@
#define __NR_sys_clock_nanosleep 1256
#define __NR_sys_fstats64 1257
#define __NR_sys_stats64 1258
+#define __NR_sys_msa 1259

#ifdef __KERNEL__

-#define NR_syscalls 256 /* length of syscall table */
+#define NR_syscalls 271 /* length of syscall table */

#if !defined(__ASSEMBLY__) && !defined(ASSEMBLER)

diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/include/linux/msa.h linux-2.5-ustate/include/linux/msa.h
--- linux-2.5-import/include/linux/msa.h Thu Jan 1 10:00:00 1970
+++ linux-2.5-ustate/include/linux/msa.h Tue Jul 22 09:28:58 2003
@@ -0,0 +1,139 @@
+/*
+ * msa.h
+ * microstate accounting
+ */
+
+#ifndef _LINUX_MSA_H
+#define _LINUX_MSA_H
+#include <config/microstate.h>
+

```

```

+#include <asm/msa.h>
+
+typedef __u64 clk_t;
+
+extern clk_t msa_last_flip[];
+
+/*
+ * Tracked states
+ */
+
+enum thread_state {
+ UNKNOWN = -1,
+ ONCPU_USER,
+ ONCPU_SYS,
+ INTERRUPTIBLE_SLEEP,
+ UNINTERRUPTIBLE_SLEEP,
+ ONACTIVEQUEUE,
+ ONEXPIREDQUEUE,
+ ZOMBIE,
+ STOPPED,
+ INTERRUPTED,
+ PAGING_SLEEP,
+ FUTEX_SLEEP,
+ POLL_SLEEP,
+
+ NR_MICRO_STATES /* Must be last */
+};
+
+#define ONCPU ONCPU_USER /* for now... */
+
+/*
+ * Times are tracked for the current task in timers[],
+ * and for the current task's children in child_timers[] (accumulated at wait() time)
+ */
+struct microstates {
+ enum thread_state cur_state;
+ enum thread_state next_state;
+ int lastqueued;
+ unsigned flags;
+ clk_t last_change;
+ clk_t timers[NR_MICRO_STATES];
+ clk_t child_timers[NR_MICRO_STATES];
+};
+
+/*
+ * Values for microstates.flags
+ */
+#define QUEUE_FLIPPED (1<<0) /* Active and Expired queues were swapped */
+#define MSA_SYS (1<<1) /* this task executing in system call */
+
+/*

```

```

+ * A system call for getting the timers.
+ * The number of timers wanted is passed as argument, in case not all
+ * are needed (and to guard against when we add more timers!)
+ */
+
+#define MSA_SELF 0
+#define MSA_CHILDREN 1
+
+#if defined __KERNEL__
+extern long sys_msa(int ntimers, int which, clk_t *timers);
+#if defined(CONFIG_MICROSTATE)
+#include <asm/current.h>
+#include <asm/irq.h>
+
+#define MSA_SOFTIRQ NR_IRQS
+
+void msa_init_timer(struct task_struct *task);
+void msa_switch(struct task_struct *prev, struct task_struct *next);
+void msa_update_parent(struct task_struct *parent, struct task_struct *this);
+void msa_init(struct task_struct *p);
+void msa_set_timer(struct task_struct *p, int state);
+void msa_start_irq(int irq);
+void msa_continue_irq(int oldirq, int newirq);
+void msa_finish_irq(int irq);
+void msa_start_syscall(void);
+void msa_end_syscall(void);
+
+clk_t msa_irq_time(int cpu, int irq);
+
+#ifdef TASK_STRUCT_DEFINED
+static inline void msa_next_state(struct task_struct *p, enum thread_state next_state)
+{
+    p->microstates.next_state = next_state;
+}
+static inline void msa_flip_expired(struct task_struct *prev) {
+    prev->microstates.flags |= QUEUE_FLIPPED;
+}
+
+static inline void msa_syscall(void) {
+    if (current->microstates.cur_state == ONCPU_USER)
+        msa_start_syscall();
+    else
+        msa_end_syscall();
+}
+
+#else
+#define msa_next_state(p, s) ((p)->microstates.next_state = (s))
+#define msa_flip_expired(p) ((p)->microstates.flags |= QUEUE_FLIPPED)
+#define msa_syscall() do { \

```


Linux-Kernel: [PATCH] Microstate accounting for 2.6.0-test1

```

linux-2.5-import/kernel/Makefile linux-2.5-ustate/kernel/Makefile
---- linux-2.5-import/kernel/Makefile Fri Jun 13 09:37:26 2003
+++ linux-2.5-ustate/kernel/Makefile Wed Jul 2 10:17:54 2003
@@ -6,7 +6,8 @@
     exit.o itimer.o time.o softirq.o resource.o \
     sysctl.o capability.o ptrace.o timer.o user.o \
     signal.o sys.o kmod.o workqueue.o pid.o \
- rcupdate.o intermodule.o extable.o params.o posix-timers.o
+ rcupdate.o intermodule.o extable.o params.o posix-timers.o \
+ msa.o

obj-$(CONFIG_FUTEX) += futex.o
obj-$(CONFIG_GENERIC_ISA_DMA) += dma.o
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/kernel/exit.c linux-2.5-ustate/kernel/exit.c
---- linux-2.5-import/kernel/exit.c Tue Jul 8 08:32:12 2003
+++ linux-2.5-ustate/kernel/exit.c Tue Jul 8 08:58:19 2003
@@ -81,6 +81,9 @@
     p->parent->cmajflt += p->majflt + p->cmajflt;
     p->parent->cnswap += p->nswap + p->cnswap;
     sched_exit(p);
+
+ msa_update_parent(p->parent, p);
+
     write_unlock_irq(&tasklist_lock);
     spin_unlock(&p->proc_lock);
     proc_pid_flush(proc_dentry);
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/kernel/fork.c linux-2.5-ustate/kernel/fork.c
---- linux-2.5-import/kernel/fork.c Mon Jul 21 09:10:45 2003
+++ linux-2.5-ustate/kernel/fork.c Mon Jul 21 15:06:20 2003
@@ -792,6 +792,7 @@
@@ -792,6 +792,7 @@
#endif
     p->did_exec = 0;
     p->state = TASK_UNINTERRUPTIBLE;
+ msa_init(p);

     copy_flags(clone_flags, p);
     if (clone_flags & CLONE_IDLETASK)
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/kernel/futex.c linux-2.5-ustate/kernel/futex.c
---- linux-2.5-import/kernel/futex.c Fri Jun 13 09:37:27 2003
+++ linux-2.5-ustate/kernel/futex.c Tue Jul 8 10:48:30 2003
@@ -34,6 +34,7 @@
@@ -34,6 +34,7 @@
#include <linux/futex.h>
#include <linux/vcache.h>
#include <linux/mount.h>
+#include <linux/msa.h>

#define FUTEX_HASHBITS 8

```

Linux-Kernel: [PATCH] Microstate accounting for 2.6.0-test1

```

@@ -349,6 +350,7 @@
     * the waiter from the list.
     */
     add_wait_queue(&q.waiters, &wait);
+ msa_next_state(current, FUTEX_SLEEP);
     set_current_state(TASK_INTERRUPTIBLE);
     if (!list_empty(&q.list)) {
         unlock_futex_mm();
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/kernel/msa.c linux-2.5-ustate/kernel/msa.c
--- linux-2.5-import/kernel/msa.c Thu Jan 1 10:00:00 1970
+++ linux-2.5-ustate/kernel/msa.c Tue Jul 22 09:28:58 2003
@@ -0,0 +1,333 @@
+/*
+ * Microstate accounting.
+ * Try to account for various states much more accurately than
+ * the normal code does.
+ */
+
+
+#include <config/microstate.h>
+#include <linux/types.h>
+#include <linux/errno.h>
+#include <linux/linkage.h>
+#ifdef CONFIG_MICROSTATE
+#include <asm/irq.h>
+#include <asm/hardirq.h>
+#include <linux/sched.h>
+#include <asm/uaccess.h>
+#include <linux/jiffies.h>
+
+
+static clk_t queueflip_time[NR_CPUS];
+
+
+clk_t msa_irq_times[NR_CPUS][NR_IRQS + 1];
+clk_t msa_irq_entered[NR_CPUS][NR_IRQS + 1];
+int msa_irq_pids[NR_CPUS][NR_IRQS + 1];
+
+/*
+ * Switch from one task to another.
+ * The retiring task is coming off the processor;
+ * the new task is about to run on the processor.
+ *
+ * Update the time in both.
+ *
+ * We'll eventually account for user and sys time separately.
+ * For now, they're both accumulated into ONCPU_USER.
+ */
+void
+msa_switch(struct task_struct *prev, struct task_struct *next)
+{
+ struct microstates *msprev = &prev->microstates;

```

```

+ struct microstates *msnext = &next->microstates;
+ clk_t now;
+ enum thread_state next_state;
+ int interrupted = msprev->cur_state == INTERRUPTED;
+
+ MSA_NOW(now);
+
+ if (msprev->flags & QUEUE_FLIPPED) {
+ queueflip_time[smp_processor_id()] = now;
+ msprev->flags &= ~QUEUE_FLIPPED;
+ }
+
+ /*
+ * If the queues have been flipped,
+ * update the state as of the last flip time.
+ */
+ if (msnext->cur_state == ONEXPIREDQUEUE) {
+ msnext->cur_state = ONACTIVEQUEUE;
+ msnext->timers[ONEXPIREDQUEUE] += queueflip_time[msnext->lastqueued] - msnext->last_change;
+ msnext->last_change = queueflip_time[msnext->lastqueued];
+ }
+
+ msprev->timers[msprev->cur_state] += now - msprev->last_change;
+ msnext->timers[msnext->cur_state] += now - msnext->last_change;
+
+ /* Update states */
+ switch (msprev->next_state) {
+ case UNKNOWN: /*
+ * Infer from actual state
+ */
+ switch (prev->state) {
+ case TASK_INTERRUPTIBLE:
+ next_state = INTERRUPTIBLE_SLEEP;
+ break;
+
+ case TASK_UNINTERRUPTIBLE:
+ next_state = UNINTERRUPTIBLE_SLEEP;
+ break;
+
+ case TASK_STOPPED:
+ next_state = STOPPED;
+ break;
+
+ case TASK_ZOMBIE:
+ next_state = ZOMBIE;
+ break;
+
+ case TASK_DEAD:
+ next_state = ZOMBIE;
+ break;
+
+

```

```

+ case TASK_RUNNING:
+ next_state = ONACTIVEQUEUE;
+ break;
+
+ default:
+ next_state = UNKNOWN;
+ break;
+
+ }
+ break;
+
+ case PAGING_SLEEP: /*
+ * Sleep states are PAGING_SLEEP;
+ * others inferred from task state
+ */
+ switch(prev->state) {
+ case TASK_INTERRUPTIBLE: /* FALLTHROUGH */
+ case TASK_UNINTERRUPTIBLE:
+ next_state = PAGING_SLEEP;
+ break;
+
+ case TASK_STOPPED:
+ next_state = STOPPED;
+ break;
+
+ case TASK_ZOMBIE:
+ next_state = ZOMBIE;
+ break;
+
+ case TASK_DEAD:
+ next_state = ZOMBIE;
+ break;
+
+ case TASK_RUNNING:
+ next_state = ONACTIVEQUEUE;
+ break;
+
+ default:
+ next_state = UNKNOWN;
+ break;
+ }
+ break;
+
+ default: /* Explicitly set next state */
+ next_state = msprev->next_state;
+ msprev->next_state = UNKNOWN;
+ break;
+ }
+
+ msprev->cur_state = next_state;
+ msprev->last_change = now;

```

```

+ msprev->lastqueued = smp_processor_id();
+
+ msnext->cur_state = interrupted ? INTERRUPTED : (
+ msnext->flags & MSA_SYS ? ONCPU_SYS : ONCPU_USER);
+ msnext->last_change = now;
+}
+
+/*
+ * Initialise the struct microstates in a new task (called from copy_process())
+ */
+void msa_init(struct task_struct *p)
+{
+ struct microstates *msp = &p->microstates;
+
+ memset(msp, 0, sizeof *msp);
+ MSA_NOW(msp->last_change);
+ msp->cur_state = UNINTERRUPTIBLE_SLEEP;
+}
+
+static void inline __msa_set_timer(struct microstates *msp, int next_state)
+{
+ clk_t now;
+
+ MSA_NOW(now);
+ msp->timers[msp->cur_state] += now - msp->last_change;
+ msp->last_change = now;
+ msp->cur_state = next_state;
+}
+
+/*
+ * Time stamp an explicit state change (called, e.g., from __activate_task())
+ */
+void
+msa_set_timer(struct task_struct *p, int next_state)
+{
+ struct microstates *msp = &p->microstates;
+
+ __msa_set_timer(msp, next_state);
+ msp->lastqueued = smp_processor_id();
+ msp->next_state = UNKNOWN;
+}
+
+/*
+ * Helper routines, to be called from assembly language stubs
+ */
+void msa_start_syscall(void)
+{
+ struct microstates *msp = &current->microstates;
+
+ __msa_set_timer(msp, ONCPU_SYS);

```

```

+ msp->flags |= MSA_SYS;
+}
+
+void msa_end_syscall(void)
+{
+ struct microstates *msp = &current->microstates;
+
+ __msa_set_timer(msp, ONCPU_USER);
+ msp->flags &= ~MSA_SYS;
+}
+
+/*
+ * Accumulate child times into parent, after zombie is over.
+ */
+void msa_update_parent(struct task_struct *parent, struct task_struct *this)
+{
+ enum thread_state s;
+ clk_t *msp = parent->microstates.child_timers;
+ struct microstates *mp = &this->microstates;
+ clk_t *msc = mp->timers;
+ clk_t *msgc = mp->child_timers;
+ clk_t now;
+
+ /*
+ * State could be ZOMBIE (if parent is interested)
+ * or something else (if the parent isn't interested)
+ */
+ MSA_NOW(now);
+ msc[mp->cur_state] += now - mp->last_change;
+
+ for (s = 0; s < NR_MICRO_STATES; s++) {
+ *msp++ += *msc++ + *msgc++;
+ }
+}
+
+void msa_start_irq(int irq)
+{
+ struct task_struct *p = current;
+ struct microstates *mp = &p->microstates;
+ clk_t now;
+ int cpu = smp_processor_id();
+
+ MSA_NOW(now);
+ mp->timers[mp->cur_state] += now - mp->last_change;
+ mp->last_change = now;
+ mp->cur_state = INTERRUPTED;
+
+ msa_irq_entered[cpu][irq] = now;
+ /* DEBUGGING */
+ msa_irq_pids[cpu][irq] = current->pid;
+}

```

```

+
+void msa_continue_irq(int oldirq, int newirq)
+{
+ clk_t now;
+ int cpu = smp_processor_id();
+ MSA_NOW(now);
+
+ msa_irq_times[cpu][oldirq] += now - msa_irq_entered[cpu][oldirq];
+ msa_irq_entered[cpu][newirq] = now;
+ msa_irq_pids[cpu][newirq] = current->pid;
+}
+
+
+void msa_finish_irq(int irq)
+{
+ struct task_struct *p = current;
+ struct microstates *mp = &p->microstates;
+ clk_t now;
+ int cpu = smp_processor_id();
+
+ MSA_NOW(now);
+
+ /*
+ * Interrupts can nest.
+ * Set current state to ONCPU
+ * iff we're not in a nested interrupt.
+ */
+ if (irq_count() == 0) {
+ mp->timers[mp->cur_state] += now - mp->last_change;
+ mp->last_change = now;
+ mp->cur_state = ONCPU_USER;
+ }
+ msa_irq_times[cpu][irq] += now - msa_irq_entered[cpu][irq];
+
+}
+
+/* return interrupt handling duration in microseconds */
+clk_t msa_irq_time(int cpu, int irq)
+{
+ clk_t x = MSA_TO_NSEC(msa_irq_times[cpu][irq]);
+ do_div(x, 1000);
+ return x;
+}
+
+/*
+ * The msa system call ---- get microstate data for self or waited-for children.
+ */
+long asmlinkage sys_msa(int ntimers, int which, clk_t __user *timers)
+{
+ clk_t now;
+ clk_t *tp;

```

```

+ int i;
+ struct microstates *msp = &current->microstates;
+
+ switch (which) {
+ case MSA_SELF:
+ case MSA_CHILDREN:
+ break;
+ default:
+ return -EINVAL;
+ }
+
+ if (ntimers > NR_MICRO_STATES)
+ ntimers = NR_MICRO_STATES;
+
+ if (which == MSA_SELF) {
+ BUG_ON(msp->cur_state != ONCPU_USER);
+
+ if (ntimers > 0) {
+ MSA_NOW(now);
+ /* Should be ONCPU_SYS */
+ msp->timers[ONCPU_USER] += now - msp->last_change;
+ msp->last_change = now;
+ }
+ }
+
+ tp = which == MSA_SELF ? msp->timers : msp->child_timers;
+ for (i = 0; i < ntimers; i++) {
+ clk_t x = MSA_TO_NSEC(*tp++);
+ if (copy_to_user(timers++, &x, sizeof x))
+ return -EFAULT;
+ }
+ return 0L;
+ }
+
+ #else
+ asmlinkage long sys_msa(int ntimers, __u64 *timers)
+ {
+ return -ENOSYS;
+ }
+ #endif
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/kernel/sched.c linux-2.5-ustate/kernel/sched.c
--- linux-2.5-import/kernel/sched.c Mon Jul 21 09:10:45 2003
+++ linux-2.5-ustate/kernel/sched.c Mon Jul 21 15:06:20 2003
@@ -335,6+335,7 @@
 */
static inline void __activate_task(task_t *p, runqueue_t *rq)
{
+ msa_set_timer(p, ONACTIVEQUEUE);
  enqueue_task(p, rq->active);
  nr_running_inc(rq);

```

```

}
@@ -558,6 +559,7 @@
    if (unlikely(!current->array))
        __activate_task(p, rq);
    else {
+ msa_set_timer(p, ONACTIVEQUEUE);
        p->prio = current->prio;
        list_add_tail(&p->run_list, &current->run_list);
        p->array = current->array;
@@ -1241,6 +1243,7 @@
        if (!TASK_INTERACTIVE(p) || EXPIRED_STARVING(rq)) {
            if (!rq->expired_timestamp)
                rq->expired_timestamp = jiffies;
+ msa_next_state(p, ONEXPIREDQUEUE);
            enqueue_task(p, rq->expired);
        } else
            enqueue_task(p, rq->active);
@@ -1324,6 +1327,7 @@
        rq->expired = array;
        array = rq->active;
        rq->expired_timestamp = 0;
+ msa_flip_expired(prev);
    }

    idx = sched_find_first_bit(array->bitmap);
@@ -1339,6 +1343,8 @@
        rq->nr_switches++;
        rq->curr = next;

+ msa_switch(prev, next);
+
        prepare_arch_switch(rq, next);
        prev = context_switch(rq, prev, next);
        barrier();
@@ -1994,6 +2000,7 @@
    */
    if (likely(!rt_task(current))) {
        dequeue_task(current, array);
+ msa_next_state(current, ONEXPIREDQUEUE);
        enqueue_task(current, rq->expired);
    } else {
        list_del(&current->run_list);
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/mm/filemap.c linux-2.5-ustate/mm/filemap.c
--- linux-2.5-import/mm/filemap.c Mon Jul 14 11:12:20 2003
+++ linux-2.5-ustate/mm/filemap.c Tue Jul 22 09:28:58 2003
@@ -273,8 +273,11 @@
    do {
        prepare_to_wait(waitqueue, &wait, TASK_UNINTERRUPTIBLE);
        if (test_bit(bit_nr, &page->flags)) {
+ msa_next_state(current, PAGING_SLEEP);

```

Linux-Kernel: [PATCH] Microstate accounting for 2.6.0-test1

```

        sync_page(page);
+ msa_next_state(current, PAGING_SLEEP);
        io_schedule();
+ msa_next_state(current, UNKNOWN);
    }
    } while (test_bit(bit_nr, &page->flags));
    finish_wait(waitqueue, &wait);
diff -Nur --exclude=RCS --exclude=CVS --exclude=SCCS --exclude=BitKeeper --exclude=ChangeSet
linux-2.5-import/mm/memory.c linux-2.5-ustate/mm/memory.c
--- linux-2.5-import/mm/memory.c Mon Jul 14 11:12:20 2003
+++ linux-2.5-ustate/mm/memory.c Tue Jul 22 09:40:21 2003
@@ -1490,16 +1490,20 @@

        entry = *pte;
        if (!pte_present(entry)) {
+ int ret;
+
        /*
         * If it truly wasn't present, we know that kswapd
         * and the PTE updates will not touch it later. So
         * drop the lock.
        */
        if (pte_none(entry))
- return do_no_page(mm, vma, address, write_access, pte, pmd);
- if (pte_file(entry))
- return do_file_page(mm, vma, address, write_access, pte, pmd);
- return do_swap_page(mm, vma, address, pte, pmd, entry, write_access);
+ ret = do_no_page(mm, vma, address, write_access, pte, pmd);
+ else if (pte_file(entry))
+ ret = do_file_page(mm, vma, address, write_access, pte, pmd);
+ else
+ ret = do_swap_page(mm, vma, address, pte, pmd, entry, write_access);
+ return ret;
    }

    if (write_access) {
@@ -1532,6 +1536,7 @@
        if (is_vm_hugetlb_page(vma))
            return VM_FAULT_SIGBUS; /* mapping truncation does this. */

+ msa_next_state(current, PAGING_SLEEP);
        /*
         * We need the page table lock to synchronize with kswapd
         * and the SMP-safe atomic PTE updates.
@@ -1541,10 +1546,14 @@

        if (pmd) {
            pte_t * pte = pte_alloc_map(mm, pmd, address);
- if (pte)
- return handle_pte_fault(mm, vma, address, write_access, pte, pmd);
+ if (pte) {

```

Linux-Kernel: [PATCH] Microstate accounting for 2.6.0-test1

```
+ int ret = handle_pte_fault(mm, vma, address, write_access, pte, pmd);
+ msa_next_state(current, UNKNOWN);
+ return ret;
+ }
  }
  spin_unlock(&mm->page_table_lock);
+ msa_next_state(current, UNKNOWN);
  return VM_FAULT_OOM;
}
```

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