

# [PATCH 1/6] 2.6.16-rc1 perfmon2 patch for review

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-01/msg07977.html>

---

- *From:* Stephane Eranian <[eranian@xxxxxxxxxxxxxxxxxxxxx](mailto:eranian@xxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Fri, 20 Jan 2006 07:20:12 -0800
- 

Hello,

This a split version of the perfmon. Each chunk was split to fit the constraints of lkml on message size. the patch is relative to 2.6.16-rc1.

Chunks [1-3] represent the common part of the perfmon2 patch. This code is common to all supported architectures.

Chunk 4 represents the i386 specific perfmon2 patch. It implements the arch-specific routines for 32-bit P4/Xeon, Pentium M/P6. It also includes the 32-bit version of the PEBS sampling format.

Chunk 5 represents the x86\_64 specific perfmon2 patch. It implements the arch-specific routines for 64-bit Opteron, EM64T. It also includes the 64-bit version of the PEBS sampling format.

Chunk 6 represents the preliminary powerpc specific perfmon2 patch. It implements the arch-specific routines for the 64-bit Power 4.

The Itanium Processors (IA-64) specific patch is not posted because it is too big to be split into smaller chunks. The size comes from the fact that it needs to remove the older implementation. If you are interested, the patch can be downloaded from our project web site at:

<http://www.sf.net/projects/perfmon2>

The MIPS support is not against the same kernel tree. To avoid confusion, we did not post it directly to lkml.

The patch is submitted for review by platform maintainers.

Thanks.

```
diff -urN --exclude-from=/tmp/excl28370 linux-2.6.16-rc1.orig/Makefile linux-2.6.16-rc1/Makefile
--- linux-2.6.16-rc1.orig/Makefile 2006-01-18 08:48:14.000000000 -0800
+++ linux-2.6.16-rc1/Makefile 2006-01-18 08:50:31.000000000 -0800
@@ -557,7 +557,7 @@
```

[PATCH 1/6] 2.6.16-rc1 perfmon2 patch for review

```
ifeq ($(KBUILD_EXTMOD),)
-core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/
+core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ perfmon/

vmlinux-dirs := $(patsubst %/,%,$(filter %/, $(init-y) $(init-m) \
$(core-y) $(core-m) $(drivers-y) $(drivers-m) \
diff -urN --exclude-from=/tmp/excl28370 linux-2.6.16-rc1.orig/kernel/sched.c
linux-2.6.16-rc1/kernel/sched.c
--- linux-2.6.16-rc1.orig/kernel/sched.c 2006-01-18 08:48:28.000000000 -0800
+++ linux-2.6.16-rc1/kernel/sched.c 2006-01-18 08:50:31.000000000 -0800
@@ -49,6 +49,7 @@
#include <linux/syscalls.h>
#include <linux/times.h>
#include <linux/acct.h>
+#include <linux/perfmon.h>
#include <asm/tlb.h>

#include <asm/unistd.h>
diff -urN --exclude-from=/tmp/excl28370 linux-2.6.16-rc1.orig/kernel/sys_ni.c
linux-2.6.16-rc1/kernel/sys_ni.c
--- linux-2.6.16-rc1.orig/kernel/sys_ni.c 2006-01-18 08:48:28.000000000 -0800
+++ linux-2.6.16-rc1/kernel/sys_ni.c 2006-01-18 08:50:31.000000000 -0800
@@ -105,6 +105,19 @@
cond_syscall(sys_vm86old);
cond_syscall(sys_vm86);

+cond_syscall(sys_pfm_create_context);
+cond_syscall(sys_pfm_write_pmcs);
+cond_syscall(sys_pfm_write_pmcs);
+cond_syscall(sys_pfm_write_pmcs);
+cond_syscall(sys_pfm_read_pmcs);
+cond_syscall(sys_pfm_restart);
+cond_syscall(sys_pfm_start);
+cond_syscall(sys_pfm_stop);
+cond_syscall(sys_pfm_load_context);
+cond_syscall(sys_pfm_unload_context);
+cond_syscall(sys_pfm_create_evtsets);
+cond_syscall(sys_pfm_delete_evtsets);
+cond_syscall(sys_pfm_getinfo_evtsets);
+
/* arch-specific weak syscall entries */
cond_syscall(sys_pciconfig_read);
cond_syscall(sys_pciconfig_write);
diff -urN --exclude-from=/tmp/excl28370 linux-2.6.16-rc1.orig/lib/Makefile
linux-2.6.16-rc1/lib/Makefile
--- linux-2.6.16-rc1.orig/lib/Makefile 2006-01-02 19:21:10.000000000 -0800
+++ linux-2.6.16-rc1/lib/Makefile 2006-01-18 08:50:31.000000000 -0800
@@ -5,7 +5,7 @@
lib-y := errno.o ctype.o string.o vsprintf.o cmdline.o \
bust_spinlocks.o rbtree.o radix-tree.o dump_stack.o \
idr.o div64.o int_sqrt.o bitmap.o extable.o prio_tree.o \
- sha1.o
```

+ sha1.o carta\_random32.o

lib-y += kobject.o kref.o kobject\_uevent.o klist.o

```
diff -urN --exclude-from=/tmp/excl28370 linux-2.6.16-rc1.orig/lib/carta_random32.c
linux-2.6.16-rc1/lib/carta_random32.c
--- linux-2.6.16-rc1.orig/lib/carta_random32.c 1969-12-31 16:00:00.000000000 -0800
+++ linux-2.6.16-rc1/lib/carta_random32.c 2006-01-18 08:50:31.000000000 -0800
@@ -0,0 +1,38 @@
+/*
+ * Fast, simple, yet decent quality random number generator based on
+ * a paper by David G. Carta ("Two Fast Implementations of the
+ * `Minimal Standard' Random Number Generator," Communications of the
+ * ACM, January, 1990).
+ *
+ * Copyright (c) 2002-2005 Hewlett-Packard Development Company, L.P.
+ * Contributed by David Mosberger-Tang <davidm@xxxxxxxxxx>
+ */
+#include <linux/types.h>
+#include <linux/module.h>
+/*
+ * XXX: Hack until we figure out which header file to use.
+ * Could use linux/random.h but then we would need
+ * asm-XX/random.h which does not exists yet
+ */
+#ifdef __ia64__
+#define __HAVE_ARCH_CARTA_RANDOM32
+#endif
+
+#ifndef __HAVE_ARCH_CARTA_RANDOM32
+u32 carta_random32(u32 seed)
+{
+    #define A 16807
+    #define M ((u32) 1 << 31)
+    u64 s, prod = A * seed, p, q;
+
+    p = (prod >> 31) & (M - 1);
+    q = (prod >> 0) & (M - 1);
+    s = p + q;
+    if (s >= M)
+        s -= M - 1;
+    return s;
+}
+#else
+extern u32 carta_random32(u32 seed);
+#endif
+EXPORT_SYMBOL(carta_random32);
diff -urN --exclude-from=/tmp/excl28370 linux-2.6.16-rc1.orig/perfmon/Makefile
linux-2.6.16-rc1/perfmon/Makefile
--- linux-2.6.16-rc1.orig/perfmon/Makefile 1969-12-31 16:00:00.000000000 -0800
+++ linux-2.6.16-rc1/perfmon/Makefile 2006-01-18 08:50:31.000000000 -0800
```

[PATCH 1/6] 2.6.16-rc1 perfmon2 patch for review

```
@@ -0,0 +1,7 @@
+#
+# Copyright (c) 2005-2006 Hewlett-Packard Development Company, L.P.
+# Contributed by Stephane Eranian <eranian@xxxxxxxxxx>
+#
+obj-y = perfmon.o perfmon_res.o perfmon_fmt.o perfmon_pmu.o perfmon_proc.o \
+ perfmon_syscalls.o perfmon_file.o perfmon_ctxsw.o perfmon_intr.o
+obj-$(CONFIG_PERFMON) += perfmon_dfl_smpl.o

diff -urN --exclude-from=/tmp/excl28370 linux-2.6.16-rc1.orig/include/linux/perfmon.h
linux-2.6.16-rc1/include/linux/perfmon.h
--- linux-2.6.16-rc1.orig/include/linux/perfmon.h 1969-12-31 16:00:00.000000000 -0800
+++ linux-2.6.16-rc1/include/linux/perfmon.h 2006-01-18 08:50:31.000000000 -0800
@@ -0,0 +1,911 @@
+/*
+ * Copyright (c) 2001-2006 Hewlett-Packard Development Company, L.P.
+ * Contributed by Stephane Eranian <eranian@xxxxxxxxxx>
+ */
+
+#ifndef __LINUX_PERFMON_H__
+#define __LINUX_PERFMON_H__
+
+#ifdef CONFIG_PERFMON
+
+#include <linux/file.h>
+#include <linux/seq_file.h>
+#include <linux/interrupt.h>
+
+#define PFM_MAX_HW_PMCS 256
+#define PFM_MAX_HW_PMDS 256
+#define PFM_MAX_XTRA_PMCS 64
+#define PFM_MAX_XTRA_PMDS 64
+
+#define PFM_MAX_PMCS (PFM_MAX_HW_PMCS+PFM_MAX_XTRA_PMCS)
+#define PFM_MAX_PMDS (PFM_MAX_HW_PMDS+PFM_MAX_XTRA_PMDS)
+
+/*
+ * number of elements for each type of bitvector
+ */
+#define PFM_BVSIZE(x) (((x)+BITS_PER_LONG-1) / BITS_PER_LONG)
+#define PFM_HW_PMD_BV PFM_BVSIZE(PFM_MAX_HW_PMDS)
+#define PFM_PMD_BV PFM_BVSIZE(PFM_MAX_PMDS)
+#define PFM_PMC_BV PFM_BVSIZE(PFM_MAX_PMCS)
+
+/*
+ * custom sampling buffer identifier type
+ */
+typedef unsigned char pfm_uuid_t[16];
+
+/*
+ * generic context flags
```

```

+ *
+ */
+#define PFM_FL_NOTIFY_BLOCK 0x01 /* block task on user notifications */
+#define PFM_FL_SYSTEM_WIDE 0x02 /* create a system wide context */
+#define PFM_FL_OVFL_NO_MSG 0x80 /* no overflow msgs */
+#define PFM_FL_MAP_SETS 0x10 /* event sets are remapped */
+
+/*
+ * PMC/PMD flags to use with pfm_write_pmds() or pfm_write_pmcs()
+ *
+ * reg_flags layout:
+ * bit 00-15 : generic flags
+ * bit 16-23 : arch-specific flags
+ * bit 24-31 : error codes
+ */
+#define PFM_REGFL_OVFL_NOTIFY 0x1 /* PMD: send notification on overflow */
+#define PFM_REGFL_RANDOM 0x2 /* PMD: randomize sampling interval */
+#define PFM_REGFL_NO_EMUL64 0x4 /* PMC: no 64-bit emulation for counter */
+
+/*
+ * event set flags layout:
+ * bit 00-15 : generic flags
+ * bit 16-31 : arch-specific flags
+ */
+#define PFM_SETFL_OVFL_SWITCH 0x01 /* enable switch on overflow */
+#define PFM_SETFL_TIME_SWITCH 0x02 /* switch set on timeout */
+#define PFM_SETFL_EXPL_NEXT 0x04 /* use set_id_next as the next set */
+#define PFM_SETFL_EXCL_IDLE 0x08 /* exclude idle task (syswide only) */
+
+/*
+ * PMD/PMC return flags in case of error (ignored on input)
+ *
+ * reg_flags layout:
+ * bit 00-15 : generic flags
+ * bit 16-23 : arch-specific flags
+ * bit 24-31 : error codes
+ *
+ * Those flags are used on output and must be checked in case EINVAL is
+ * returned by a command accepting a vector of values and each has a flag
+ * field, such as pfarg_pmc_t or pfarg_pmd_t.
+ */
+#define PFM_REG_RETFL_NOTAVAIL (1<<31) /* implemented but not available */
+#define PFM_REG_RETFL_EINVAL (1<<30) /* entry is invalid */
+#define PFM_REG_RETFL_NOSET (1<<29) /* event set does not exist */
+#define PFM_REG_RETFL_MASK (PFM_REG_RETFL_NOTAVAIL|\
+ PFM_REG_RETFL_EINVAL|\
+ PFM_REG_RETFL_NOSET)
+
+/*
+#define PFM_REG_HAS_ERROR(flag) (((flag) & PFM_REG_RETFL_MASK) != 0)
+
+ */

```

```

+ * Request structure used to define a context
+ */
+typedef struct {
+ pfm_uuid_t ctx_smpl_buf_id; /* which buffer format to use */
+ u32 ctx_flags; /* noblock/block */
+ int ctx_fd; /* ret arg: fd for context */
+ size_t ctx_smpl_buf_size; /* ret arg: actual buffer size */
+ u64 ctx_reserved3[4]; /* for future use */
+} pfarg_ctx_t;
+
+/*
+ * argument structure for pfm_write_pmcs()
+ */
+typedef struct {
+ u16 reg_num; /* which register */
+ u16 reg_set; /* event set for this register */
+ u32 reg_flags; /* input: flags, return: reg error */
+ u64 reg_value; /* pmc value */
+ u64 reg_reserved2[4]; /* for future use */
+} pfarg_pmc_t;
+
+/*
+ * argument structure for pfm_write_pmds() and pfm_read_pmds()
+ */
+typedef struct {
+ u16 reg_num; /* which register */
+ u16 reg_set; /* event set for this register */
+ u32 reg_flags; /* input: flags, return: reg error */
+ u64 reg_value; /* initial pmc/pmd value */
+ u64 reg_long_reset; /* value to reload after notification */
+ u64 reg_short_reset; /* reset after counter overflow */
+ u64 reg_last_reset_val; /* return: PMD last reset value */
+ u64 reg_ovfl_switch_cnt; /* #overflows before switch */
+ unsigned long reg_reset_pmds[PFM_PMD_BV]; /* reset on overflow */
+ unsigned long reg_smpl_pmds[PFM_PMD_BV]; /* record in sample */
+ u64 reg_smpl_eventid; /* opaque event identifier */
+ u64 reg_random_mask; /* bitmask used to limit random value */
+ u32 reg_random_seed; /* seed for randomization */
+ u32 reg_reserved2[7]; /* for future use */
+} pfarg_pmd_t;
+
+/*
+ * optional argument to pfm_start(), pass NULL if no arg needed
+ */
+typedef struct {
+ u16 start_set; /* event set to start with */
+ u16 start_reserved1; /* for future use */
+ u32 start_reserved2; /* for future use */
+ u64 reserved3[3]; /* for future use */
+} pfarg_start_t;
+

```

```

+/*
+ * argument to pfm_load_context()
+ */
+typedef struct {
+ pid_t load_pid; /* thread to attach to */
+ u16 load_set; /* set to load first */
+ u16 load_reserved1; /* for future use */
+ u64 load_reserved2[3]; /* for future use */
+} pfarg_load_t;
+
+/*
+ * argument to pfm_create_evtsets()/pfm_delete_evtsets()
+ */
+typedef struct {
+ u16 set_id; /* which set */
+ u16 set_id_next; /* next set to go to */
+ u32 set_flags; /* input: flags, return: err flag */
+ struct timespec set_timeout; /* req/eff switch timeout */
+ off_t set_mmap_offset; /* ret arg: cookie for mmap offset */
+ u64 reserved[5]; /* for future use */
+} pfarg_setdesc_t;
+
+/*
+ * argument to pfm_getinfo_evtsets()
+ */
+typedef struct {
+ u16 set_id; /* which set */
+ u16 set_id_next; /* out: next set to go to */
+ u32 set_flags; /* out: flags or error */
+ unsigned long set_ovfl_pmds[PFM_PMD_BV]; /* out: last ovfl PMDs */
+ u64 set_runs; /* out: #times the set was active */
+ struct timespec set_timeout; /* out: effective switch timeout */
+ u64 set_act_duration; /* out: time set active (cycles) */
+ off_t set_mmap_offset; /* cookie to for mmap offset */
+ u64 reserved[4]; /* for future use */
+} pfarg_setinfo_t;
+
+/*
+ * default value for the user and group security parameters in
+ * /proc/sys/kernel/perfmon
+ */
+#define PFM_GROUP_PERM_ANY -1 /* any user/group */
+
+/*
+ * remapped set view
+ *
+ * IMPORTANT: cannot be bigger than PAGE_SIZE
+ */
+typedef struct {
+ u32 set_status; /* set status: active/inact */
+ u32 set_reserved1; /* for future use */

```

```

+ u64 set_runs; /* number of activations */
+ u64 set_pmds[PFM_MAX_PMDS]; /* 64-bit value of PMDS */
+ volatile unsigned long set_seq; /* sequence number of updates */
+} pfm_set_view_t;
+
+/*
+ * pfm_set_view_t status flags
+ */
+#define PFM_SETVFL_ACTIVE 0x1 /* set is active */
+
+typedef struct {
+ u32 msg_type; /* generic message header */
+ pid_t msg_ovfl_pid; /* process id */
+ unsigned long msg_ovfl_pmds[PFM_HW_PMD_BV]; /* overflowed PMDs */
+ u16 msg_active_set; /* active set at overflow */
+ u16 msg_ovfl_cpu; /* cpu of PMU interrupt */
+ pid_t msg_ovfl_tid; /* kernel thread id */
+ unsigned long msg_ovfl_ip; /* instr ptr on PMU intr */
+} pfm_ovfl_msg_t;
+
+#define PFM_MSG_OVFL 1 /* an overflow happened */
+#define PFM_MSG_END 2 /* task to which context was attached ended */
+
+typedef union {
+ u32 type;
+ pfm_ovfl_msg_t pfm_ovfl_msg;
+} pfm_msg_t;
+
+/*
+ * perfmon version number
+ */
+#define PFM_VERSION_MAJ 2U
+#define PFM_VERSION_MIN 2U
+#define PFM_VERSION (((PFM_VERSION_MAJ&0xffff)<<16)|\
+ (PFM_VERSION_MIN & 0xffff))
+#define PFM_VERSION_MAJOR(x) (((x)>>16) & 0xffff)
+#define PFM_VERSION_MINOR(x) ((x) & 0xffff)
+
+
+#ifdef __KERNEL__
+
+struct pfm_ovfl_arg {
+ u16 ovfl_pmd; /* index of overflowed PMD */
+ u16 active_set; /* set active at the time of the overflow */
+ u32 ovfl_ctrl; /* control flags */
+ u64 pmd_last_reset; /* last reset value of overflowed PMD */
+ u64 smpl_pmds_values[PFM_MAX_PMDS]; /* values of other PMDs */
+ u64 pmd_eventid; /* eventid associated with PMD */
+ u16 num_smpl_pmds; /* number of PMDS in smpl_pmd_values */
+};
+

```

```

+
+/*
+ * ovfl_ctrl bitmask of flags
+ */
+#define PFM_OVFL_CTRL_NOTIFY 0x1 /* notify user */
+#define PFM_OVFL_CTRL_RESET 0x2 /* reset overflowed pmds */
+#define PFM_OVFL_CTRL_MASK 0x4 /* mask monitoring */
+
+
+/*
+ * describe the content of the pfm_syst_info field
+ */
+#define PFM_CPUINFO_TIME_SWITCH 0x20 /* current set is time-switched */
+
+struct pfm_sysctl {
+ int debug; /* debugging via syslog */
+ int debug_ovfl; /* overflow handling debugging */
+ int expert_mode; /* PMC/PMD value checking */
+ gid_t sys_group; /* gid to create a syswide context */
+ gid_t task_group; /* gid to create a per-task context */
+ size_t arg_size_max; /* maximum vector argument size */
+ size_t smpl_buf_size_max; /* max buf mem, -1 for infinity */
+};
+
+
+DECLARE_PER_CPU(struct task_struct *, pmu_owner);
+DECLARE_PER_CPU(struct pfm_context *, pmu_ctx);
+DECLARE_PER_CPU(unsigned long, pfm_syst_info);
+DECLARE_PER_CPU(u64, pmu_activation_number);
+DECLARE_PER_CPU(struct pfm_stats, pfm_stats);
+
+/*
+ * perfmon context state
+ */
+#define PFM_CTX_UNLOADED 1 /* context is not loaded onto any task */
+#define PFM_CTX_LOADED 2 /* context is loaded onto a task */
+#define PFM_CTX_MASKED 3 /* context is loaded, monitoring is masked */
+#define PFM_CTX_ZOMBIE 4 /* context lost owner but is still attached */
+
+/*
+ * depth of message queue
+ */
+#define PFM_MAX_MSGS 8
+#define PFM_CTXQ_EMPTY(g) ((g)->ctx_msgq_head == (g)->ctx_msgq_tail)
+
+/*
+ * type of PMD reset for pfm_reset_pmds() or pfm_switch_sets()
+ */
+#define PFM_PMD_RESET_NONE 0 /* do not reset (pfm_switch_set) */
+#define PFM_PMD_RESET_SHORT 1 /* use short reset value */
+#define PFM_PMD_RESET_LONG 2 /* use long reset value */

```

```

+
+/*
+ * debugging
+ */
+#define PFM_DEBUGGING
+#ifdef PFM_DEBUGGING
+#define DPRINT(a) \
+ do { \
+ if (unlikely(pfm_sysctl.debug >0)) {\
+ printk("%s.%d: CPU%d [%d] ", __FUNCTION__, __LINE__,\
+ smp_processor_id(), current->pid); printk a; } \
+ } while (0)
+
+#define DPRINT_ovfl(a) \
+ do { \
+ if (unlikely(pfm_sysctl.debug_ovfl >0))\
+ {\
+ printk("%s.%d: CPU%d [%d] ",\
+ __FUNCTION__, __LINE__,\
+ smp_processor_id(),\
+ current->pid); printk a; } \
+ } while (0)
+#else
+#define DPRINT(a) do { } while(0)
+#define DPRINT_ovfl(a) do { } while(0)
+#endif
+
+/*
+ * global information about all sessions
+ * mostly used to synchronize between system wide and per-process
+ */
+struct pfm_sessions {
+ u32 pfs_task_sessions; /* #num loaded per-thread sessions */
+ u32 pfs_sys_sessions; /* #num loaded system wide sessions */
+ size_t pfs_cur_smpl_buf_mem; /* current smpl buf mem usage */
+ cpumask_t pfs_sys_cpumask; /* bitmask of used cpus */
+};
+
+/*
+ * PMD description structure
+ * software maintained value is in the pfm_set_view_t structure.
+ */
+struct pfm_pmd {
+ u64 lval; /* last reset value */
+ u64 ovflsw_thres; /* #overflows left before switching */
+ u64 long_reset; /* reset value on sampling overflow */
+ u64 short_reset; /* reset value on overflow */
+ unsigned long reset_pmds[PFM_PMD_BV]; /* pmds to reset on overflow */
+ unsigned long smpl_pmds[PFM_PMD_BV]; /* pmds to record on overflow */
+ u64 mask; /* mask for generator */

```

```

+ u64 seed; /* seed for generator (must be 64 bits here) */
+ u32 flags; /* notify/do not notify */
+ u64 ovflsw_ref_thres; /* #overflows before switching to next set */
+ u64 eventid; /* overflow event identifier */
+};
+
+/*
+ * perfmon context: encapsulates all the state of a monitoring session
+ */
+struct pfm_event_set {
+ u16 set_id;
+ u16 set_id_next; /* which set to go to from this one */
+ u32 set_flags; /* public set flags */
+
+ struct pfm_event_set *set_next; /* next in the ordered list */
+ struct pfm_event_set *set_switch_next; /* address of set to go to */
+ u32 set_priv_flags; /* private flags */
+ u32 set_npend_ovfls; /* number of pending PMD overflow */
+
+ unsigned long set_used_pmds[PFM_PMD_BV]; /* used PMDs */
+ unsigned long set_povfl_pmds[PFM_PMD_BV]; /* pending overflowed PMDs */
+ unsigned long set_ovfl_pmds[PFM_PMD_BV]; /* last overflowed PMDs */
+ unsigned long set_reset_pmds[PFM_PMD_BV]; /* PMDs to reset */
+ unsigned long set_ovfl_notify[PFM_PMD_BV]; /* notify on overflow */
+ u64 set_pmc[PFM_MAX_PMCS]; /* PMC values */
+
+ u16 set_nused_pmds; /* max number of used PMDs */
+ u16 set_nused_pmc; /* max number of used PMCs */
+
+ struct pfm_pmd set_pmds[PFM_MAX_PMDS]; /* 64-bit SW PMDs */
+ pfm_set_view_t *set_view; /* pointer to view */
+ u64 set_switch_timeout; /* switch timeout */
+ u64 set_timeout; /* timeout remaining */
+ u64 set_duration_start; /* start cycles */
+ u64 set_duration; /* total active cycles */
+ off_t set_mmap_offset; /* view mmap offset */
+ unsigned long set_used_pmc[PFM_PMC_BV]; /* used PMCs (keep for arbitration) */
+};
+
+/*
+ * common private event set flags (set_priv_flags)
+ *
+ * upper 16 bits: for arch-specific use
+ * lower 16 bits: for common use
+ */
+#define PFM_SETFL_PRIV_MOD_PMDS 0x1 /* PMD register(s) modified */
+#define PFM_SETFL_PRIV_MOD_PMCS 0x2 /* PMC register(s) modified */
+#define PFM_SETFL_PRIV_SWITCH 0x4 /* must switch set on restart */
+#define PFM_SETFL_PRIV_MOD_BOTH (PFM_SETFL_PRIV_MOD_PMDS |
PFM_SETFL_PRIV_MOD_PMCS)
+

```

```

+/*
+ * context flags
+ */
+struct pfm_context_flags {
+ unsigned int block:1; /* task blocks on user notifications */
+ unsigned int system:1; /* do system wide monitoring */
+ unsigned int excl_idle:1; /* exclude idle task (syswide) */
+ unsigned int trap_reason:2; /* reason for pfm_handle_work() */
+ unsigned int no_msg:1; /* no message sent on overflow */
+ unsigned int can_restart:1; /* allowed to issue a PFM_RESTART */
+ unsigned int switch_ovfl:1; /* switch set on counter ovfl */
+ unsigned int switch_time:1; /* switch set on timeout */
+ unsigned int mapset:1; /* event sets are remapped */
+ unsigned int started:1; /* pfm_start() issued */
+ unsigned int reserved:21; /* for future use */
+};
+
+/*
+ * values for trap_reason
+ */
+#define PFM_TRAP_REASON_NONE 0x0 /* nothing to do */
+#define PFM_TRAP_REASON_BLOCK 0x1 /* block on overflow */
+#define PFM_TRAP_REASON_RESET 0x2 /* reset PMDs */
+#define PFM_TRAP_REASON_ZOMBIE 0x3 /* cleanup because of ZOMBIE */
+
+/*
+ * perfmon context: encapsulates all the state of a monitoring session
+ */
+struct pfm_smpl_fmt;
+struct pfm_context {
+ spinlock_t ctx_lock; /* context protection */
+ u8 pad[128-sizeof(spinlock_t)];
+
+ struct file *ctx_filp; /* filp */
+
+ struct pfm_context_flags ctx_flags; /* flags */
+ u32 ctx_state; /* state */
+ struct task_struct *ctx_task; /* attached task */
+
+ struct completion ctx_restart_complete; /* block on notification */
+ u64 ctx_duration_start; /* last cycles at last activation */
+ u64 ctx_duration; /* total cycles context was active */
+ u64 ctx_last_act; /* last activation */
+ u32 ctx_last_cpu; /* last CPU used (SMP only) */
+ u32 ctx_cpu; /* cpu bound to context */
+ struct pfm_smpl_fmt *ctx_smpl_fmt; /* buffer format callbacks */
+ void *ctx_smpl_addr; /* smpl buffer base */
+ size_t ctx_smpl_size;
+ wait_queue_head_t ctx_msgq_wait;
+ pfm_msg_t ctx_msgq[PFM_MAX_MSGS];
+ int ctx_msgq_head;

```

```

+ int ctx_msgq_tail;
+ struct fasync_struct *ctx_async_queue;
+
+ struct pfm_event_set *ctx_active_set; /* active set */
+ struct pfm_event_set *ctx_sets; /* ordered list of sets */
+
+ /*
+ * save stack space by allocating temporary variables for
+ * pfm_overflow_handler() in pfm_context
+ */
+ struct pfm_ovfl_arg ovfl_arg;
+ unsigned long ovfl_ovfl_notify[PFM_PMD_BV];
+ };
+
+ #define ctx_fl_block ctx_flags.block
+ #define ctx_fl_system ctx_flags.system
+ #define ctx_fl_trap_reason ctx_flags.trap_reason
+ #define ctx_fl_no_msg ctx_flags.no_msg
+ #define ctx_fl_can_restart ctx_flags.can_restart
+ #define ctx_fl_mapset ctx_flags.mapset
+ #define ctx_fl_intr_pending ctx_flags.intr_pending
+ #define ctx_fl_started ctx_flags.started
+
+ #define pfm_ctx_arch(c) ((struct pfm_arch_context *)((c)+1))
+
+ typedef int (*fmt_validate_t)(u32 flags, u16 npmds, void *arg);
+ typedef int (*fmt_getsize_t)(u32 flags, void *arg, size_t *size);
+ typedef int (*fmt_init_t)(struct pfm_context *ctx, void *buf, u32 flags, u16 npmds, void *arg);
+ typedef int (*fmt_restart_t)(int is_active, u32 *ovfl_ctrl, void *buf);
+ typedef int (*fmt_exit_t)(void *buf);
+ typedef int (*fmt_handler_t)(void *buf, struct pfm_ovfl_arg *arg,
+ unsigned long ip, u64 stamp, void *data);
+
+ struct pfm_smpl_fmt {
+ char *fmt_name; /* name of the format (required) */
+ pfm_uuid_t fmt_uuid; /* 128-bit unique id (required) */
+ size_t fmt_arg_size; /* size of fmt args for ctx create */
+ u32 fmt_flags; /* format specific flags */
+
+ fmt_validate_t fmt_validate; /* validate context flags */
+ fmt_getsize_t fmt_getsize; /* get size for sampling buffer */
+ fmt_init_t fmt_init; /* initialize buffer area */
+ fmt_handler_t fmt_handler; /* overflow handler (required) */
+ fmt_restart_t fmt_restart; /* restart after notification */
+ fmt_exit_t fmt_exit; /* context termination */
+
+ struct list_head fmt_list; /* internal use only */
+
+ struct module *owner; /* pointer to module owner */
+ u32 fmt_qdepth; /* Max notify queue depth (required) */
+ };

```

```

+
+#define PFM_FMTFL_IS_BUILTIN 0x1 /* fmt is compiled in */
+
+#ifdef MODULE
+#define PFM_PMU_BUILTIN_FLAG 0 /* not built as a module */
+#else
+#define PFM_PMU_BUILTIN_FLAG PFM_PMUFL_IS_BUILTIN /* built as a module */
+#endif
+
+static inline void pfm_set_pmu_owner(struct task_struct *task,
+ struct pfm_context *ctx)
+{
+ BUG_ON(task && task->pid == 0);
+ __get_cpu_var(pmu_owner) = task;
+ __get_cpu_var(pmu_ctx) = ctx;
+}
+
+#ifdef CONFIG_SMP
+static inline void pfm_inc_activation(void)
+{
+ __get_cpu_var(pmu_activation_number)++;
+}
+static inline void pfm_set_activation(struct pfm_context *ctx)
+{
+ ctx->ctx_last_act = __get_cpu_var(pmu_activation_number);
+}
+static inline void pfm_set_last_cpu(struct pfm_context *ctx, int cpu)
+{
+ ctx->ctx_last_cpu = cpu;
+}
+#else
+#define pfm_inc_activation() do { } while(0)
+#define pfm_set_activation(c) do { } while(0)
+#define pfm_set_last_cpu(c, p) do { } while(0)
+#endif
+
+static inline void pfm_modview_begin(struct pfm_event_set *set)
+{
+ set->set_view->set_seq++;
+}
+
+static inline void pfm_modview_end(struct pfm_event_set *set)
+{
+ set->set_view->set_seq++;
+}
+
+static inline void pfm_retflag_set(u32 flags, u32 val)
+{
+ flags &= ~PFM_REG_RETFL_MASK;
+ flags |= (val);
+}

```

```

+
+/*
+ * exported to support backward compatibility mode (IA-64)
+ */
+extern int pfm_compat_update_pmd(struct pfm_context *, u16, u16,
+ u32, u64 *, u64 *, u64);
+
+
+extern struct pfm_pmu_config *pfm_pmu_conf;
+extern struct pfm_sysctl pfm_sysctl;
+
+extern int pfm_get_args(void __user *arg, size_t sz, void **kargs);
+extern int pfm_get_smpl_arg(pfm_uuid_t uuid, void *uaddr, size_t usz,
+ void **arg, struct pfm_smpl_fmt **);
+
+extern void pfm_undo_create_context(int fd, struct pfm_context *ctx);
+extern int pfm_alloc_fd(struct file **cfile);
+extern void pfm_syswide_cleanup_other_cpu(struct pfm_context *ctx);
+
+extern int __pfm_write_pmcs(struct pfm_context *, pfarg_pmc_t *, int);
+extern int __pfm_write_pmds(struct pfm_context *, pfarg_pmd_t *, int, int);
+extern int __pfm_read_pmds(struct pfm_context *, pfarg_pmd_t *, int);
+extern void __pfm_reset_stats(void);
+extern int __pfm_load_context(struct pfm_context *ctx, pfarg_load_t *req);
+extern int __pfm_unload_context(struct pfm_context *ctx);
+extern int __pfm_stop(struct pfm_context *ctx);
+extern int __pfm_restart(struct pfm_context *ctx);
+extern int __pfm_start(struct pfm_context *ctx, pfarg_start_t *start);
+extern int __pfm_delete_evtsets(struct pfm_context *ctx, void *arg, int count);
+extern int __pfm_getinfo_evtsets(struct pfm_context *ctx, pfarg_setinfo_t *req,
+ int count);
+extern int __pfm_create_evtsets(struct pfm_context *ctx, pfarg_setdesc_t *req,
+ int count);
+extern int __pfm_create_context(pfarg_ctx_t *, struct pfm_smpl_fmt *, void *, int,
+ struct pfm_context **);
+extern int pfm_check_task_state(struct pfm_context *ctx, int cmd,
+ unsigned long *flags);
+
+extern struct pfm_event_set *pfm_find_set(struct pfm_context *ctx, u16 set_id,
+ int alloc);
+
+extern struct pfm_context * pfm_get_ctx(int fd);
+
+extern void pfm_context_free(struct pfm_context *ctx);
+extern struct pfm_context *pfm_context_alloc(void);
+
+extern int pfm_register_pmu_config(struct pfm_pmu_config *cfg);
+extern void pfm_unregister_pmu_config(struct pfm_pmu_config *cfg);
+
+extern int pfm_pmu_conf_get(void);
+extern void pfm_pmu_conf_put(void);

```

```

+
+extern int pfm_reserve_session(struct pfm_context *ctx, u32 cpu);
+extern int pfm_release_session(struct pfm_context *ctx, u32 cpu);
+
+extern int pfm_smpl_buffer_alloc(struct pfm_context *ctx, size_t rsize);
+extern int pfm_reserve_buf_space(size_t size);
+extern void pfm_release_buf_space(size_t size);
+
+extern struct pfm_smpl_fmt *pfm_smpl_fmt_get(pfm_uid_t uuid);
+extern void pfm_smpl_fmt_put(struct pfm_smpl_fmt *fmt);
+extern int pfm_use_smpl_fmt(pfm_uid_t uuid);
+
+extern int pfm_proc_init(void);
+extern void pfm_proc_show_sessions(struct seq_file *);
+extern void pfm_proc_show_pmu_map(struct seq_file *, void *);
+extern void pfm_proc_show_fmt(struct seq_file *m);
+
+extern int pfm_register_smpl_fmt(struct pfm_smpl_fmt *fmt);
+extern int pfm_unregister_smpl_fmt(pfm_uid_t uuid);
+extern irqreturn_t pfm_interrupt_handler(int, void *, struct pt_regs *);
+extern void pfm_save_pmds_release(struct pfm_context *ctx);
+
+extern void pfm_reset_pmds(struct pfm_context *ctx, struct pfm_event_set *set,
+ int reset_mode);
+extern void pfm_switch_sets(struct pfm_context *ctx,
+ struct pfm_event_set *new_set,
+ int reset_mode,
+ int no_restart);
+
+extern void pfm_mask_monitoring(struct pfm_context *ctx);
+extern int pfm_ovfl_notify_user(struct pfm_context *ctx,
+ struct pfm_event_set *set,
+ unsigned long ip);
+
+extern int init_pfm_fs(void);
+extern int pfm_is_fd(struct file *filp);
+
+extern u32 carta_random32 (u64 seed);
+
+extern void __pfm_exit_thread(struct task_struct *);
+extern void __pfm_copy_thread(struct task_struct *task);
+extern void __pfm_ctxswin(struct task_struct *task);
+extern void __pfm_ctxswout(struct task_struct *task);
+extern void __pfm_handle_work(void);
+extern void __pfm_handle_switch_timeout(void);
+
+extern void pfm_resume_after_ovfl(struct pfm_context *ctx);
+
+static inline void pfm_put_ctx(struct pfm_context *ctx)
+{
+ fput(ctx->ctx_filp);

```

```

+}
+
+#define PFM_MAX_NUM_SETS 65536
+#define PFM_SET_REMAP_SCALAR PAGE_SIZE
+#define PFM_SET_REMAP_OFFS 16384 /* number of pages to offset */
+#define PFM_SET_REMAP_BASE (PFM_SET_REMAP_OFFS*PAGE_SIZE)
+#define PFM_SET_REMAP_OFFS_MAX (PFM_SET_REMAP_OFFS+\
+ PFM_MAX_NUM_SETS*PFM_SET_REMAP_SCALAR)
+
+
+#define PMC_IS_LAST(c,i) ((c)->pmc_desc[i].type & PFM_REG_END)
+#define PMD_IS_LAST(c,i) ((c)->pmd_desc[i].type & PFM_REG_END)
+
+/*
+ * This structure is finalized at boot time and contains
+ * a description of the PMU main characteristics.
+ *
+ * probe_pmu routine return value:
+ * - 1 means recognized PMU
+ * - 0 means not recognized PMU
+ */
+typedef int (*pfm_reg_check_t)(struct pfm_context *ctx,
+ struct pfm_event_set *set, u16 cnum, u32 flags, u64 *val);
+
+typedef u64 (*pfm_pmd_sread_t)(struct pfm_context *ctx, unsigned int cnum);
+typedef void (*pfm_pmd_swrite_t)(struct pfm_context *ctx, unsigned int cnum, u64 val);
+typedef void (*pfm_pmc_swrite_t)(struct pfm_context *ctx, unsigned int cnum, u64 val);
+
+/*
+ * PMU description
+ *
+ * fields marked as "rt" are computed at runtime when module is inserted
+ */
+struct pfm_reg_desc;
+struct pfm_pmu_config {
+ unsigned long impl_pmcs[PFM_PMC_BV]; /* rt: impl PMC */
+ unsigned long impl_pmds[PFM_PMD_BV]; /* rt: impl PMD */
+ unsigned long impl_rw_pmds[PFM_PMD_BV]; /* rt: impl RW PMD */
+ unsigned long cnt_pmds[PFM_PMD_BV]; /* rt: impl counter */
+ u64 ovfl_mask; /* rt: overflow mask */
+ u16 max_pmc; /* rt: highest+1 impl PMC */
+ u16 max_pmd; /* rt: highest+1 impl PMD */
+ u16 max_rw_pmd; /* rt: highest+1 impl RW PMD */
+ u16 first_cnt_pmd; /* rt: first counting PMD */
+ u16 max_cnt_pmd; /* rt: highest+1 impl counter */
+ u16 num_pmcs; /* rt: logical PMCS */
+ u16 num_pmds; /* rt: logical PMDS */
+ u16 num_counters; /* rt: PMC/PMD counter pairs */
+
+ char *pmu_name; /* PMU family name */
+ char *version; /* config module version number */

```

```

+ int counter_width; /* width of hardware counter */
+ struct pfm_reg_desc *pmc_desc; /* PMC register descriptions */
+ struct pfm_reg_desc *pmd_desc; /* PMD register descriptions */
+ pfm_reg_check_t pmc_write_check; /* PMC write checker callback */
+ pfm_reg_check_t pmd_write_check; /* PMD write checker callback */
+ pfm_reg_check_t pmd_read_check; /* PMD read checker callback */
+ pfm_pmd_sread_t pmd_sread; /* PMD model specific read */
+ pfm_pmd_swrite_t pmd_swrite; /* PMD model specific write */
+ pfm_pmc_swrite_t pmc_swrite; /* PMC model specific write */
+ int (*probe_pmu)(void); /* probe PMU routine */
+ void *arch_info; /* arch-specific information */
+ u32 flags; /* set of flags */
+ struct module *owner; /* pointer to module struct */
+};
+
+/*
+ * pmu_config flags
+ */
+#define PFM_PMUFL_IS_BUILTIN 0x1 /* pmu config is compiled in */
+
+/*
+ * generic information about a PMC or PMD register
+ */
+struct pfm_reg_desc {
+ u16 type; /* role of the register */
+ char *desc; /* HW register description string */
+ u64 default_value; /* power-on default value (quiet) */
+ u64 reserved_mask; /* reserved bits: 0 means reserved */
+ u64 no_emul64_mask; /* bits to clear for PFM_REGFL_NO_EMUL64 */
+};
+
+
+/*
+ * type of a PMU register (16-bit bitmask) for use with pfm_reg_desc.type
+ */
+#define PFM_REG_NA 0x0 /* not available (not impl. or no access) */
+#define PFM_REG_I 0x1 /* implemented */
+#define PFM_REG_END 0x2 /* end marker */
+#define PFM_REG_RC 0x4 /* PMD: has read_checker */
+#define PFM_REG_WC 0x8 /* has write_checker */
+#define PFM_REG_C64 0x10 /* PMD: 64-bit virtualization */
+#define PFM_REG_RO 0x20 /* PMD: read-only (writes ignored) */
+#define PFM_REG_V 0x40 /* PMD: virtual reg implemented by PMU description */
+#define PFM_REG_NO64 0x80 /* PMC: supports REGFL_NOEMUL64 */
+
+
+/*
+ * define some shortcuts for common types
+ */
+#define PFM_REG_W (PFM_REG_WC|PFM_REG_I)
+#define PFM_REG_W64 (PFM_REG_WC|PFM_REG_NO64|PFM_REG_I)
+#define PFM_REG_C (PFM_REG_C64|PFM_REG_I)

```

```

+
+#define PFM_ONE_64 ((u64)1)
+
+struct pfm_stats {
+ u64 pfm_replay_intr_count; /* replayed ovfl interrupts */
+ u64 pfm_real_intr_count; /* processed ovfl interrupts */
+ u64 pfm_all_intr_count; /* total ovfl interrupts */
+ u64 pfm_intr_cycles; /* cycles in ovfl interrupts */
+ u64 pfm_intr_phase1; /* cycles in ovfl interrupts */
+ u64 pfm_intr_phase2; /* cycles in ovfl interrupts */
+ u64 pfm_intr_phase3; /* cycles in ovfl interrupts */
+ u64 pfm_intr_cycles_min; /* min cycles in ovfl interrupts */
+ u64 pfm_intr_cycles_max; /* max cycles in ovfl interrupts */
+ u64 pfm_smpl_handler_calls; /* # calls smpl buffer handler */
+ u64 pfm_smpl_handler_cycles; /* cycle in smpl format handler */
+ u64 pfm_switch_count; /* #set_switches on this CPU */
+ u64 pfm_switch_cycles; /* cycles for switching sets */
+ u64 pfm_handle_timeout_count; /* #count of set timeouts handled */
+};
+
+#include <asm/perfmon.h>
+
+/*
+ * check_mask bitmask values for pfm_check_task_state()
+ */
+#define PFM_CMD_STOPPED 0x01 /* command needs thread stopped */
+#define PFM_CMD_UNLOADED 0x02 /* command needs ctx unloaded */
+#define PFM_CMD_UNLOAD 0x04 /* command is unload */
+
+#if BITS_PER_LONG == 32
+#define PFM_BPL 32
+#define PFM_LBPL 5 /* log2(BPL) */
+#elif BITS_PER_LONG == 64
+#define PFM_BPL 64
+#define PFM_LBPL 6 /* log2(BPL) */
+#else
+#error "you need to define log2(BITS_PER_LONG)"
+#endif
+
+static inline void pfm_bv_set(unsigned long *bv, unsigned int rnum)
+{
+ bv[rnum>>PFM_LBPL] |= 1UL << (rnum&(PFM_BPL-1));
+}
+
+static inline int pfm_bv_isset(unsigned long *bv, unsigned int rnum)
+{
+ return bv[rnum>>PFM_LBPL] & (1UL <<(rnum&(PFM_BPL-1))) ? 1 : 0;
+}
+
+static inline void pfm_bv_clear(unsigned long *bv, unsigned int rnum)
+{

```

```

+ bv[rnum>>PFM_LBPL] &= ~(1UL << (rnum&(PFM_BPL-1)));
+}
+
+/*
+ * for for special, not necessarily PMU-related, PMD registers implemented
+ * for PMU model or HW platform. Some of those PMDs may give access to some
+ * SW resources.
+ */
+static inline u64 pfm_read_pmd(struct pfm_context *ctx, unsigned int cnum)
+{
+ if (pfm_pmu_conf->pmd_desc[cnum].type & PFM_REG_V)
+ return pfm_pmu_conf->pmd_sread(ctx, cnum);
+
+ return pfm_arch_read_pmd(ctx, cnum);
+}
+
+static inline void pfm_write_pmd(struct pfm_context *ctx, unsigned int cnum, u64 value)
+{
+ /*
+ * PMD writes are ignored for read-only registers
+ */
+ if (pfm_pmu_conf->pmd_desc[cnum].type & PFM_REG_RO)
+ return;
+
+ if (pfm_pmu_conf->pmd_desc[cnum].type & PFM_REG_V) {
+ pfm_pmu_conf->pmd_swrite(ctx, cnum, value);
+ return;
+ }
+ pfm_arch_write_pmd(ctx, cnum, value);
+}
+
+static inline void pfm_exit_thread(struct task_struct *task)
+{
+ if (task->thread.pfm_context)
+ __pfm_exit_thread(task);
+}
+
+static inline void pfm_handle_work(void)
+{
+ if (current->thread.pfm_context)
+ __pfm_handle_work();
+}
+
+static inline void pfm_copy_thread(struct task_struct *task,
+ struct pt_regs *regs)
+{
+ /*
+ * task+regs are child state
+ */
+ if (task->thread.pfm_context)
+ __pfm_copy_thread(task);

```

```

+}
+
+static inline void pfm_ctxswin(struct task_struct *task)
+{
+ /*
+ * cannot check for pfm_context because
+ * not necessarily present in system-wide
+ */
+ __pfm_ctxswin(task);
+}
+
+static inline void pfm_ctxswout(struct task_struct *task)
+{
+ /*
+ * cannot check for pfm_context because
+ * not necessarily present in system-wide
+ */
+ __pfm_ctxswout(task);
+}
+
+static inline void pfm_handle_switch_timeout(void)
+{
+ unsigned long info;
+ info = __get_cpu_var(pfm_syst_info);
+ if (info & PFM_CPUINFO_TIME_SWITCH)
+ __pfm_handle_switch_timeout();
+}
+
+extern void pfm_vector_init(void);
+
+#define pfm_sample_thread(t)
+#define pfm_release_task(t)
+#define pfm_set_cpus_allowed(p, m)
+#endif /* __KERNEL__ */
+#else /* !CONFIG_PERFMON */
+
+#ifdef __KERNEL__
+
+#define pfm_exit_thread(_t) do { } while (0)
+#define pfm_handle_work() do { } while (0)
+#define pfm_copy_thread(_t, _r) do { } while (0)
+#define pfm_ctxswin(_t) do { } while (0)
+#define pfm_ctxswout(_t) do { } while (0)
+#define pfm_handle_switch_timeout() do { } while (0)
+#define pfm_release_task(_t) do { } while (0)
+#define pfm_set_cpus_allowed(_t, _m) do { } while (0)
+#define pfm_sample_thread(_t) do { } while (0)
+#define pfm_vector_init() do { } while (0)
+
+ /* use for IA-64 only */
+#ifdef __ia64__

```

```

+#define pfm_release_dbregs(_t) do { } while (0)
+#define pfm_use_dbregs(_t) (0)
+#endif
+
+#endif
+#endif /* CONFIG_PERFMON */
+
+#endif /* __LINUX_PERFMON_H__ */
diff -urN --exclude-from=/tmp/excl28370 linux-2.6.16-rc1.orig/include/linux/perfmon_dfl_smpl.h
linux-2.6.16-rc1/include/linux/perfmon_dfl_smpl.h
--- linux-2.6.16-rc1.orig/include/linux/perfmon_dfl_smpl.h 1969-12-31 16:00:00.000000000 -0800
+++ linux-2.6.16-rc1/include/linux/perfmon_dfl_smpl.h 2006-01-18 08:50:31.000000000 -0800
@@ -0,0 +1,77 @@
+/*
+ * Copyright (c) 2005-2006 Hewlett-Packard Development Company, L.P.
+ * Contributed by Stephane Eranian <eranian@xxxxxxxxxx>
+ *
+ * This file implements the new dfl sampling buffer format
+ * for perfmon2 subsystem.
+ */
+#ifndef __PERFMON_DFL_SMPL_H__
+#define __PERFMON_DFL_SMPL_H__ 1
+
+#define PFM_DFL_SMPL_UUID { \
+ 0xd1, 0x39, 0xb2, 0x9e, 0x62, 0xe8, 0x40, 0xe4,\
+ 0xb4, 0x02, 0x73, 0x07, 0x87, 0x92, 0xe9, 0x37}
+
+/*
+ * format specific parameters (passed at context creation)
+ */
+typedef struct {
+ size_t buf_size; /* size of the buffer in bytes */
+ u32 flags; /* buffer specific flags */
+ u32 res1; /* for future use */
+ u64 reserved[2]; /* for future use */
+} pfm_dfl_smpl_arg_t;
+
+/*
+ * combined context+format specific structure. Can be passed
+ * to pfm_context_create()
+ */
+typedef struct {
+ pfarg_ctx_t ctx_arg;
+ pfm_dfl_smpl_arg_t buf_arg;
+} pfm_dfl_smpl_ctx_arg_t;
+
+/*
+ * This header is at the beginning of the sampling buffer returned to the user.
+ * It is directly followed by the first record.
+ */
+typedef struct {

```

[PATCH 1/6] 2.6.16-rc1 perfmon2 patch for review

```
+ size_t hdr_count; /* how many valid entries */
+ size_t hdr_cur_offs; /* current offset from top of buffer */
+ u64 hdr_overflows; /* #overflows for buffer */
+ size_t hdr_buf_size; /* bytes in the buffer */
+ size_t hdr_min_buf_space; /* minimal buffer size (internal use) */
+ u32 hdr_version; /* smpl format version */
+ u32 hdr_reserved1; /* for future use */
+ u64 hdr_reserved[10]; /* for future use */
+} pfm_dfl_smpl_hdr_t;
+
+/*
+ * Entry header in the sampling buffer. The header is directly followed
+ * with the values of the PMD registers of interest saved in increasing
+ * index order: PMD4, PMD5, and so on. How many PMDs are present depends
+ * on how the session was programmed.
+ *
+ * In the case where multiple counters overflow at the same time, multiple
+ * entries are written consecutively.
+ *
+ * last_reset_value member indicates the initial value of the overflowed PMD.
+ */
+typedef struct {
+ pid_t pid; /* thread id (for NPTL, this is gettid()) */
+ u16 ovfl_pmd; /* index of overflowed PMD for this sample */
+ u16 reserved; /* for future use */
+ u64 last_reset_val; /* initial value of overflowed PMD */
+ unsigned long ip; /* where did the overflow interrupt happened */
+ u64 tstamp; /* overflow timetamp */
+ u16 cpu; /* cpu on which the overflow occured */
+ u16 set; /* event set active when overflow occurred */
+ pid_t tgid; /* thread group id (for NPTL, this is getpid())*/
+} pfm_dfl_smpl_entry_t;
+
+#define PFM_DFL_SMPL_VERSION_MAJ 1U
+#define PFM_DFL_SMPL_VERSION_MIN 0U
+#define PFM_DFL_SMPL_VERSION (((PFM_DFL_SMPL_VERSION_MAJ&0xffff)<<16)|\
+ (PFM_DFL_SMPL_VERSION_MIN & 0xffff))
+
+#endif /* __PERFMON_DFL_SMPL_H__ */
```

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

- 
- *Follow-Ups:*
    - ◆ **Re: [PATCH 1/6] 2.6.16-rc1 perfmon2 patch for review**  
◇ From: Bryan O'Sullivan

[PATCH 1/6] 2.6.16-rc1 perfmon2 patch for review

- Prev by Date: [\[PATCH 6/6\] 2.6.16-rc1 perfmon2 patch for review](#)
- Next by Date: [\[PATCH 3/6\] 2.6.16-rc1 perfmon2 patch for review](#)
- Previous by thread: [\[PATCH 6/6\] 2.6.16-rc1 perfmon2 patch for review](#)
- Next by thread: [\*\*Re: \[PATCH 1/6\] 2.6.16-rc1 perfmon2 patch for review\*\*](#)
- Index(es):
  - ◆ [Date](#)
  - ◆ [Thread](#)