Subject: Re: [PATCH 1/8] Resource counters
Posted by Andrew Morton on Wed, 30 May 2007 21:44:56 GMT
View Forum Message <> Reply to Message

On Wed, 30 May 2007 19:26:34 +0400
Pavel Emelianov <xemul@openvz.org> wrote:

> Introduce generic structures and routines for resource accounting.
>
> Each resource accounting container is supposed to aggregate it,
> container_subsystem_state and its resource-specific members within.
>
> Signed-off-by: Pavel Emelianov <xemul@openvz.org>
>
> ---
>
> diff -upr linux-2.6.22-rc2-mm1.orig/include/linux/res_counter.h
linux-2.6.22-rc2-mm1-0/include/linux/res_counter.h
> --- linux-2.6.22-rc2-mm1.orig/include/linux/res_counter.h 2007-05-30 16:16:58.000000000
+0400
> +++ linux-2.6.22-rc2-mm1-0/include/linux/res_counter.h 2007-05-30 16:13:09.000000000 +0400
> @@ -0,0 +1,83 @@
> +#ifndef __RES_COUNTER_H__
> +#define __RES_COUNTER_H__
> +/*
> + * resource counters
> + *
> + * Copyright 2007 OpenVZ SWsoft Inc
> + *
> + * Author: Pavel Emelianov <xemul@openvz.org>
> + *
> + */
> +
> +#include <linux/container.h>
> +
> +struct res_counter {
> + unsigned long usage;
> + unsigned long limit;
> + unsigned long failcnt;
> + spinlock_t lock;
> +};

The one place where documentation really pays off is on data structures.
Please document your core data structures with great care.

> +enum {
> + RES_USAGE,
> + RES_LIMIT,

```
> + RES_FAILCNT,
> +};
```

Documenting these wouild be good too.

```
> +ssize_t res_counter_read(struct res_counter *cnt, int member,
> +  const char __user *buf, size_t nbytes, loff_t *pos);
> +ssize_t res_counter_write(struct res_counter *cnt, int member,
> +  const char __user *buf, size_t nbytes, loff_t *pos);
> +
> +static inline void res_counter_init(struct res_counter *cnt)
> +{
> + spin_lock_init(&cnt->lock);
> + cnt->limit = (unsigned long)LONG_MAX;
> +}
> +
> +static inline int res_counter_charge_locked(struct res_counter *cnt,
> +  unsigned long val)
> +{
> + if (cnt->usage <= cnt->limit - val) {
> +  cnt->usage += val;
> +  return 0;
> + }
> +
> + cnt->failcnt++;
> + return -ENOMEM;
> +}
> +
> +static inline int res_counter_charge(struct res_counter *cnt,
> +  unsigned long val)
> +{
> + int ret;
> + unsigned long flags;
> +
> + spin_lock_irqsave(&cnt->lock, flags);
> + ret = res_counter_charge_locked(cnt, val);
> + spin_unlock_irqrestore(&cnt->lock, flags);
> + return ret;
> +}
> +
> +static inline void res_counter_uncharge_locked(struct res_counter *cnt,
> +  unsigned long val)
> +{
> + if (unlikely(cnt->usage < val)) {
> +  WARN_ON(1);
> +  val = cnt->usage;
> + }
> +
```

> + cnt->usage -= val;
> +}
> +
> +static inline void res_counter_uncharge(struct res_counter *cnt,
> + unsigned long val)
> +{
> + unsigned long flags;
> +
> + spin_lock_irqsave(&cnt->lock, flags);
> + res_counter_uncharge_locked(cnt, val);
> + spin_unlock_irqrestore(&cnt->lock, flags);
> +}

The above functions will expand to a truly astonishing amount of code. I
suspect they all should be uninlined.

> +#endif
> diff -upr linux-2.6.22-rc2-mm1.orig/init/Kconfig linux-2.6.22-rc2-mm1-0/init/Kconfig
> --- linux-2.6.22-rc2-mm1.orig/init/Kconfig 2007-05-30 16:13:08.000000000 +0400
> +++ linux-2.6.22-rc2-mm1-0/init/Kconfig 2007-05-30 16:13:09.000000000 +0400
> @@ -328,6 +328,10 @@ config CPUSETS
>
>   Say N if unsure.
>
> +config RESOURCE_COUNTERS
> + bool
> + select CONTAINERS
> +
> config SYSFS_DEPRECATED
>   bool "Create deprecated sysfs files"
>   default y
> diff -upr linux-2.6.22-rc2-mm1.orig/kernel/Makefile linux-2.6.22-rc2-mm1-0/kernel/Makefile
> --- linux-2.6.22-rc2-mm1.orig/kernel/Makefile 2007-05-30 16:13:08.000000000 +0400
> +++ linux-2.6.22-rc2-mm1-0/kernel/Makefile 2007-05-30 16:13:09.000000000 +0400
> @@ -56,6 +56,7 @@ obj-$(CONFIG_SYSCTL) += utsname_sysctl.o
> obj-$(CONFIG_UTS_NS) += utsname.o
> obj-$(CONFIG_TASK_DELAY_ACCT) += delayacct.o
> obj-$(CONFIG_TASKSTATS) += taskstats.o tsacct.o
> +obj-$(CONFIG_RESOURCE_COUNTERS) += res_counter.o
>
> ifneq ($(CONFIG_SCHED_NO_NO_OMIT_FRAME_POINTER),y)
> # According to Alan Modra <alan@linuxcare.com.au>, the -fno-omit-frame-pointer is
> diff -upr linux-2.6.22-rc2-mm1.orig/kernel/res_counter.c
linux-2.6.22-rc2-mm1-0/kernel/res_counter.c
> --- linux-2.6.22-rc2-mm1.orig/kernel/res_counter.c 2007-05-30 16:16:58.000000000 +0400
> +++ linux-2.6.22-rc2-mm1-0/kernel/res_counter.c 2007-05-30 16:13:09.000000000 +0400
> @@ -0,0 +1,73 @@
> +/*

> + * resource containers
> + *
> + * Copyright 2007 OpenVZ SWsoft Inc
> + *
> + * Author: Pavel Emelianov <xemul@openvz.org>
> + *
> + */
> +
> +#include <linux/types.h>
> +#include <linux/parser.h>
> +#include <linux/fs.h>
> +#include <linux/res_counter.h>
> +#include <asm/uaccess.h>

We typically put a blank line between the linux/ includes adn the asm/
includes.

If include/linux/foo.h exists, please include that rather than
include/asm/foo.h

> +static inline unsigned long *res_counter_member(struct res_counter *cnt, int member)
> +{
> + switch (member) {
> + case RES_USAGE:
> + return &cnt->usage;
> + case RES_LIMIT:
> + return &cnt->limit;
> + case RES_FAILCNT:
> + return &cnt->failcnt;
> + };
> +
> + BUG();
> + return NULL;
> +}
> +
> +ssize_t res_counter_read(struct res_counter *cnt, int member,
> + const char __user *userbuf, size_t nbytes, loff_t *pos)
> +{
> + unsigned long *val;
> + char buf[64], *s;
> +
> + s = buf;
> + val = res_counter_member(cnt, member);
> + s += sprintf(s, "%lu\n", *val);
> + return simple_read_from_buffer((void __user *)userbuf, nbytes,
> +   pos, buf, s - buf);
> +}
> +

```
> +ssize_t res_counter_write(struct res_counter *cnt, int member,
> +  const char __user *userbuf, size_t nbytes, loff_t *pos)
> +{
> + int ret;
> + char *buf, *end;
> + unsigned long tmp, *val;
> +
> + buf = kmalloc(nbytes + 1, GFP_KERNEL);
> + ret = -ENOMEM;
> + if (buf == NULL)
> +  goto out;
> +
> + buf[nbytes] = 0;
> + ret = -EFAULT;
> + if (copy_from_user(buf, userbuf, nbytes))
> +  goto out_free;
> +
> + ret = -EINVAL;
> + tmp = simple_strtoul(buf, &end, 10);
> + if (*end != '\0')
> +  goto out_free;
> +
> + val = res_counter_member(cnt, member);
> + *val = tmp;
> + ret = nbytes;
> +out_free:
> + kfree(buf);
> +out:
> + return ret;
> +}
```

You'd think that by now the kernel would have a "get a decimal number from userspace" library function. But I don't think we do.
