
Subject: Re: [-mm PATCH 1/9] Memory controller resource counters (v6)

Posted by [Paul Menage](#) on Tue, 11 Sep 2007 00:42:20 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi Balbir/Pavel,

As I mentioned to you directly at the kernel summit, I think it might be cleaner to integrate resource counters more closely with control groups. So rather than controllers such as the memory controller having to create their own boilerplate cf_type structures and read/write functions, it should be possible to just call a function something like

```
control_group_add_rescounter(struct cgroup *cg, struct cgroup_subsys *ss,
                           struct res_counter *res,
                           const char *name)
```

and have it handle all the userspace API. This would simplify the task of keeping a consistent userspace API between different controllers using the resource counter abstraction.

Paul

On 8/17/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> From: Pavel Emelianov <xemul@openvz.org>

>

> 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>

> Signed-off-by: <balbir@linux.vnet.ibm.com>

> ---

>

> include/linux/res_counter.h | 102 ++++++++++++++++++++++++++++++++++++++

> init/Kconfig | 7 ++

> kernel/Makefile | 1

> kernel/res_counter.c | 120 ++++++++++++++++++++++++++++++++++++++

> 4 files changed, 230 insertions(+)

>

> diff -puN /dev/null include/linux/res_counter.h

> --- /dev/null 2007-06-01 20:42:04.000000000 +0530

> +++ linux-2.6.23-rc2-mm2-balbir/include/linux/res_counter.h 2007-08-17 13:14:18.000000000 +0530

> @@ -0,0 +1,102 @@

> +#ifndef __RES_COUNTER_H__

```

> + #define __RES_COUNTER_H__
> +
> + /*
> + * Resource Counters
> + * Contain common data types and routines for resource accounting
> + *
> + * Copyright 2007 OpenVZ SWsoft Inc
> + *
> + * Author: Pavel Emelianov <xemul@openvz.org>
> + *
> + */
> +
> + #include <linux/container.h>
> +
> + /*
> + * The core object. the container that wishes to account for some
> + * resource may include this counter into its structures and use
> + * the helpers described beyond
> + */
> +
> + struct res_counter {
> +     /*
> +      * the current resource consumption level
> +      */
> +     unsigned long usage;
> +     /*
> +      * the limit that usage cannot exceed
> +      */
> +     unsigned long limit;
> +     /*
> +      * the number of unsuccessful attempts to consume the resource
> +      */
> +     unsigned long failcnt;
> +     /*
> +      * the lock to protect all of the above.
> +      * the routines below consider this to be IRQ-safe
> +      */
> +     spinlock_t lock;
> + };
> +
> + /*
> + * Helpers to interact with userspace
> + * res_counter_read/_write - put/get the specified fields from the
> + * res_counter struct to/from the user
> + *
> + * @counter:   the counter in question
> + * @member:   the field to work with (see RES_xxx below)
> + * @buf:      the buffer to operate on,...

```

```

> + * @nbytes: its size...
> + * @pos:    and the offset.
> + */
> +
> + ssize_t res_counter_read(struct res_counter *counter, int member,
> +     const char __user *buf, size_t nbytes, loff_t *pos);
> + ssize_t res_counter_write(struct res_counter *counter, int member,
> +     const char __user *buf, size_t nbytes, loff_t *pos);
> +
> + /*
> + * the field descriptors. one for each member of res_counter
> + */
> +
> + enum {
> +     RES_USAGE,
> +     RES_LIMIT,
> +     RES_FAILCNT,
> + };
> +
> + /*
> + * helpers for accounting
> + */
> +
> + void res_counter_init(struct res_counter *counter);
> +
> + /*
> + * charge - try to consume more resource.
> + *
> + * @counter: the counter
> + * @val: the amount of the resource. each controller defines its own
> + *       units, e.g. numbers, bytes, Kbytes, etc
> + *
> + * returns 0 on success and <0 if the counter->usage will exceed the
> + * counter->limit _locked call expects the counter->lock to be taken
> + */
> +
> + int res_counter_charge_locked(struct res_counter *counter, unsigned long val);
> + int res_counter_charge(struct res_counter *counter, unsigned long val);
> +
> + /*
> + * uncharge - tell that some portion of the resource is released
> + *
> + * @counter: the counter
> + * @val: the amount of the resource
> + *
> + * these calls check for usage underflow and show a warning on the console
> + * _locked call expects the counter->lock to be taken
> + */

```

```

> +
> +void res_counter_uncharge_locked(struct res_counter *counter, unsigned long val);
> +void res_counter_uncharge(struct res_counter *counter, unsigned long val);
> +
> +#endif
> diff -puN init/Kconfig~res_counters_infra init/Kconfig
> --- linux-2.6.23-rc2-mm2/init/Kconfig~res_counters_infra      2007-08-17 13:14:18.000000000
+0530
> +++ linux-2.6.23-rc2-mm2-balbir/init/Kconfig  2007-08-17 13:14:18.000000000 +0530
> @@ -337,6 +337,13 @@ config CPUSETS
>
>     Say N if unsure.
>
> +config RESOURCE_COUNTERS
> +     bool "Resource counters"
> +     help
> +         This option enables controller independent resource accounting
> +         infrastructure that works with containers
> +     depends on CONTAINERS
> +
> config SYSFS_DEPRECATED
>     bool "Create deprecated sysfs files"
>     default y
> diff -puN kernel/Makefile~res_counters_infra kernel/Makefile
> --- linux-2.6.23-rc2-mm2/kernel/Makefile~res_counters_infra  2007-08-17 13:14:18.000000000
+0530
> +++ linux-2.6.23-rc2-mm2-balbir/kernel/Makefile 2007-08-17 13:14:18.000000000 +0530
> @@ -58,6 +58,7 @@ obj-$(CONFIG_RELAY) += relay.o
> obj-$(CONFIG_SYSCTL) += utsname_sysctl.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 -puN /dev/null kernel/res_counter.c
> --- /dev/null  2007-06-01 20:42:04.000000000 +0530
> +++ linux-2.6.23-rc2-mm2-balbir/kernel/res_counter.c  2007-08-17 13:14:18.000000000 +0530
> @@ -0,0 +1,120 @@
> +/*
> + * 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 <linux/uaccess.h>
> +
> + void res_counter_init(struct res_counter *counter)
> + {
> +     spin_lock_init(&counter->lock);
> +     counter->limit = (unsigned long)LONG_MAX;
> + }
> +
> + int res_counter_charge_locked(struct res_counter *counter, unsigned long val)
> + {
> +     if (counter->usage > (counter->limit - val)) {
> +         counter->failcnt++;
> +         return -ENOMEM;
> +     }
> +
> +     counter->usage += val;
> +     return 0;
> + }
> +
> + int res_counter_charge(struct res_counter *counter, unsigned long val)
> + {
> +     int ret;
> +     unsigned long flags;
> +
> +     spin_lock_irqsave(&counter->lock, flags);
> +     ret = res_counter_charge_locked(counter, val);
> +     spin_unlock_irqrestore(&counter->lock, flags);
> +     return ret;
> + }
> +
> + void res_counter_uncharge_locked(struct res_counter *counter, unsigned long val)
> + {
> +     if (WARN_ON(counter->usage < val))
> +         val = counter->usage;
> +
> +     counter->usage -= val;
> + }
> +
> + void res_counter_uncharge(struct res_counter *counter, unsigned long val)
> + {
> +     unsigned long flags;
> +
> +     spin_lock_irqsave(&counter->lock, flags);
> +     res_counter_uncharge_locked(counter, val);

```

```

> + spin_unlock_irqrestore(&counter->lock, flags);
> +}
> +
> +
> +static inline unsigned long *res_counter_member(struct res_counter *counter,
> +                                              int member)
> +{
> +    switch (member) {
> +    case RES_USAGE:
> +        return &counter->usage;
> +    case RES_LIMIT:
> +        return &counter->limit;
> +    case RES_FAILCNT:
> +        return &counter->failcnt;
> +    };
> +
> +    BUG();
> +    return NULL;
> +}
> +
> +ssize_t res_counter_read(struct res_counter *counter, 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(counter, 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 *counter, 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(counter, member);
> +      *val = tmp;
> +      ret = nbytes;
> +out_free:
> +      kfree(buf);
> +out:
> +      return ret;
> +}
> _
>
> --
> Warm Regards,
> Balbir Singh
> Linux Technology Center
> IBM, ISTL
>
> --
> To unsubscribe, send a message with 'unsubscribe linux-mm' in
> the body to majordomo@kvack.org. For more info on Linux MM,
> see: http://www.linux-mm.org/ .
> Don't email: <a href="mailto:dont@kvack.org"> email@kvack.org </a>
>
```

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
