Subject: Re: [-mm PATCH 1/8] Memory controller resource counters (v2)
Posted by Dave Hansen on Fri, 06 Jul 2007 17:24:01 GMT
View Forum Message <> Reply to Message

On Thu, 2007-07-05 at 22:20 -0700, Balbir Singh wrote:
> +/*
> + * 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
> + */

I'm going to nitpick a bit here. Nothing major, I promise. ;)

Could we make these comments into nice sentences with capitalization? I think it makes them easier to read in long comments.

How about something like this for the comment:

```
/*
 * A container wishing to account for a resource should include this
 * structure into one of its own.  It may use the helpers below.
 */
```

The one above is worded a little bit strangely.

```
> +struct res_counter {
> + /*
> +  * the current resource consumption level
> +  */
> + unsigned long usage;
> + /*
> +  * the limit that usage cannot exceed
> +  */
> + unsigned long limit;
> + /*
> +  * the number of insuccessful attempts to consume the resource
> +  */
```

unsuccessful

```
> + unsigned long failcnt;
> + /*
> +  * the lock to protect all of the above.
> +  * the routines below consider this to be IRQ-safe
> +  */
> + spinlock_t lock;
> +};
```

Do we really need all of these comments? Some of them are a wee bit
self-explanatory. I think we mostly know what a limit is. ;)

```
> +/*
> + * helpers to interact with userspace
> + * res_counter_read/_write - put/get the specified fields from the
> + * res_counter struct to/from the user
> + *
> + * @cnt:     the counter in question
> + * @member:  the field to work with (see RES_xxx below)
> + * @buf:     the buffer to opeate on,...
> + * @nbytes:  its size...
> + * @pos:     and the offset.
> + */
> +
> +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);
> +
> +/*
> + * 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 *cnt);
> +
> +/*
> + * charge - try to consume more resource.
> + *
> + * @cnt: 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 cnt->usage will exceed the cnt->limit
> + * _locked call expects the cnt->lock to be taken
> + */
> +
> +int res_counter_charge_locked(struct res_counter *cnt, unsigned long val);
```

> +int res_counter_charge(struct res_counter *cnt, unsigned long val);
> +
> +/*
> + * uncharge - tell that some portion of the resource is released
> + *
> + * @cnt: 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 cnt->lock to be taken
> + */
> +
> +void res_counter_uncharge_locked(struct res_counter *cnt, unsigned long val);
> +void res_counter_uncharge(struct res_counter *cnt, unsigned long val);
> +
> +#endif
> diff -puN init/Kconfig~res_counters_infra init/Kconfig
> --- linux-2.6.22-rc6/init/Kconfig~res_counters_infra 2007-07-05 13:45:17.000000000 -0700
> +++ linux-2.6.22-rc6-balbir/init/Kconfig 2007-07-05 13:45:17.000000000 -0700
> @@ -320,6 +320,10 @@ config CPUSETS
>
>   Say N if unsure.
>
> +config RESOURCE_COUNTERS
> + bool
> + select CONTAINERS
> +
> config SYSFS_DEPRECATED
>   bool "Create deprecated sysfs files"
>   default y
> diff -puN kernel/Makefile~res_counters_infra kernel/Makefile
> --- linux-2.6.22-rc6/kernel/Makefile~res_counters_infra 2007-07-05 13:45:17.000000000 -0700
> +++ linux-2.6.22-rc6-balbir/kernel/Makefile 2007-07-05 13:45:17.000000000 -0700
> @@ -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 08:12:04.000000000 -0700
> +++ linux-2.6.22-rc6-balbir/kernel/res_counter.c 2007-07-05 13:45:17.000000000 -0700
> @@ -0,0 +1,121 @@
> +/*
> + * 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 *cnt)
> +{
> + spin_lock_init(&cnt->lock);
> + cnt->limit = (unsigned long)LONG_MAX;
> +}
> +
> +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;
> +}

More nitpicking...

Can we leave the normal control flow in the lowest indentation level,
and have only errors in the indented if(){} blocks? Something like
this:

> +int res_counter_charge_locked(struct res_counter *cnt, unsigned long
val)
> +{
> + if (cnt->usage > cnt->limit - val) {
> + cnt->failcnt++;
> + return -ENOMEM;
> + }
> + cnt->usage += val;
> + return 0;
> +}

Also, can you do my poor brain a favor an expand "cnt" to "counter"?
You're not saving _that_ much typing ;)

```
> +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;
> +}
> +
> +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;
> +}
```

It actually looks like the WARN_ON() macros "return" values. You should
be able to:

```
 if (WARN_ON(cnt->usage < val))
  val = count->usage;
```

```
> +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);
> +}
> +
> +
> +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);
> +}

Why do we need that cast?

> +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);

Do we need some checking on nbytes? Is it sanitized before it gets
here?

> + ret = -ENOMEM;
> + if (buf == NULL)
> + goto out;
> +
> + buf[nbytes] = 0;

Please use '\0'. 0 isn't a char.

> + 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;
> +}
> _
>
-- Dave

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