
Subject: [PATCH 0/3] Consolidate cgroup files creation for resource counters (v2)

Posted by [Pavel Emelianov](#) on Thu, 04 Oct 2007 09:18:01 GMT

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

Changes from previous version

- * made the names configurable
- * fixed race between res_counter_populate and reading any of these files (memset to zero could spoof the pointers)

Right now we have only one controller in -mm tree - the memory one - and it initializes all its files manually. After I developed the kernel memory one it turned out, that the names of resource counter specific files has changed. In the future, if anything in the resource counters change, we'll have to rewrite all the controllers we will have by that time.

To make all the containers working with the resource counters look similar I propose to create the resource counters specific files in one place.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: [PATCH 1/3] Typedefs the read and write functions in cftype

Posted by [Pavel Emelianov](#) on Thu, 04 Oct 2007 09:18:55 GMT

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

This is just to reduce the code amount in the future.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h

index 8747932..0635004 100644

--- a/include/linux/cgroup.h

+++ b/include/linux/cgroup.h

@@ -178,6 +178,15 @@ struct css_set {

* - the 'cftype' of the file is file->f_dentry->d_fsdata

*/

+struct cftype;

+

```

+typedef ssize_t (*cft_read) (struct cgroup *cont, struct cftype *cft,
+ struct file *file,
+ char __user *buf, size_t nbytes, loff_t *ppos);
+typedef ssize_t (*cft_write) (struct cgroup *cont, struct cftype *cft,
+ struct file *file,
+ const char __user *buf, size_t nbytes, loff_t *ppos);
+
#define MAX_CFTYPE_NAME 64
struct cftype {
    /* By convention, the name should begin with the name of the
@@ -185,18 +194,14 @@ struct cftype {
    char name[MAX_CFTYPE_NAME];
    int private;
    int (*open) (struct inode *inode, struct file *file);
- ssize_t (*read) (struct cgroup *cont, struct cftype *cft,
- struct file *file,
- char __user *buf, size_t nbytes, loff_t *ppos);
+ cft_read read;
    /*
     * read_uint() is a shortcut for the common case of returning a
     * single integer. Use it in place of read()
     */
    u64 (*read_uint) (struct cgroup *cont, struct cftype *cft);
- ssize_t (*write) (struct cgroup *cont, struct cftype *cft,
- struct file *file,
- const char __user *buf, size_t nbytes, loff_t *ppos);

+ cft_write write;
    /*
     * write_uint() is a shortcut for the common case of accepting
     * a single integer (as parsed by simple_strtoull) from

```

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

Subject: [PATCH 2/3] Introduce the res_counter_populate() function
Posted by [Pavel Emelianov](#) on Thu, 04 Oct 2007 09:20:55 GMT
[View Forum Message](#) <> [Reply to Message](#)

This one is responsible for initializing the RES_CFT_MAX files properly and register them inside the container.

The caller must provide the cgroup, the cgroup_subsys, the RES_CFT_MAX * sizeof(cftype) chunk of zeroed memory, the units of measure and the read and write callbacks.

Right now I made names for two units - bytes and items. Maybe later we will add more (pages, HZ?).

In the future, if we add more res_counter files, change their names or anything else, no caller will be affected.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

diff --git a/include/linux/res_counter.h b/include/linux/res_counter.h

index 61363ce..bee93c3 100644

--- a/include/linux/res_counter.h

+++ b/include/linux/res_counter.h

```
@@ -58,6 +58,17 @@ ssize_t res_counter_write(struct res_cou
    const char __user *buf, size_t nbytes, loff_t *pos,
    int (*write_strategy)(char *buf, unsigned long long *val));
```

```
+enum {
```

```
+ RES_UNITS_BYTES,
```

```
+ RES_UNITS_ITEMS,
```

```
+
```

```
+ RES_UNITS_MAX
```

```
+};
```

```
+
```

```
+int res_counter_populate(struct cgroup_subsys *ss, struct cgroup *cont,
```

```
+ struct cftype files[], int units,
```

```
+ cft_read read_fn, cft_write write_fn);
```

```
+
```

```
/*
```

```
 * the field descriptors. one for each member of res_counter
```

```
 */
```

```
@@ -66,6 +77,8 @@ enum {
```

```
    RES_USAGE,
```

```
    RES_LIMIT,
```

```
    RES_FAILCNT,
```

```
+
```

```
+ RES_CFT_MAX,
```

```
};
```

```
/*
```

diff --git a/kernel/res_counter.c b/kernel/res_counter.c

index d7f43cd..ae77b6e 100644

--- a/kernel/res_counter.c

+++ b/kernel/res_counter.c

```
@@ -10,9 +10,45 @@
```

```
#include <linux/types.h>
```

```

#include <linux/parser.h>
#include <linux/fs.h>
#include <linux/cgroup.h>
#include <linux/res_counter.h>
#include <linux/uaccess.h>

+static char * units_names[RES_UNITS_MAX][RES_CFT_MAX] = {
+ [RES_UNITS_BYTES] = {
+  "usage_in_bytes",
+  "limit_in_bytes",
+  "failcnt",
+ },
+ [RES_UNITS_ITEMS] = {
+  "usage",
+  "limit",
+  "failcnt",
+ },
+};
+
+static void cft_init(struct cftype files[], int type, int units,
+ cft_read read_fn, cft_write write_fn)
+{
+ if (files[type].name[0] == '\0') {
+  strcpy(files[type].name, units_names[units][type]);
+  files[type].private = type;
+  files[type].read = read_fn;
+  files[type].write = write_fn;
+ }
+}
+
+int res_counter_populate(struct cgroup_subsys *ss, struct cgroup *cont,
+ struct cftype files[], int units,
+ cft_read read_fn, cft_write write_fn)
+{
+ cft_init(files, RES_USAGE, units, read_fn, NULL);
+ cft_init(files, RES_LIMIT, units, read_fn, write_fn);
+ cft_init(files, RES_FAILCNT, units, read_fn, NULL);
+
+ return cgroup_add_files(cont, ss, files, RES_CFT_MAX);
+}
+
+void res_counter_init(struct res_counter *counter)
+{
+ spin_lock_init(&counter->lock);

```

Containers mailing list
Containers@lists.linux-foundation.org

Subject: [PATCH 3/3] Use the res_counter_populate in memory controller

Posted by [Pavel Emelianov](#) on Thu, 04 Oct 2007 09:21:37 GMT

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

Note, that the controller code dealing with the cftype files for resource counters becomes much shorter and won't have to be changed in the future.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 1b8bf24..2e62d24 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -519,28 +519,14 @@ static ssize_t mem_control_type_read(str
```

```
static struct cftype mem_cgroup_files[] = {
{
- .name = "usage_in_bytes",
- .private = RES_USAGE,
- .read = mem_cgroup_read,
- },
- {
- .name = "limit_in_bytes",
- .private = RES_LIMIT,
- .write = mem_cgroup_write,
- .read = mem_cgroup_read,
- },
- {
- .name = "failcnt",
- .private = RES_FAILCNT,
- .read = mem_cgroup_read,
- },
- {
- .name = "control_type",
- .write = mem_control_type_write,
- .read = mem_control_type_read,
- },
};
```

```
+static struct cftype mem_res_counter_files[RES_CFT_MAX];
+
static struct mem_cgroup init_mem_cgroup;
```

```

static struct cgroup_subsys_state *
@@ -574,6 +560,13 @@ static void mem_cgroup_destroy(struct cg
static int mem_cgroup_populate(struct cgroup_subsys *ss,
    struct cgroup *cont)
{
+ int ret;
+
+ ret = res_counter_populate(ss, cont, mem_res_counter_files,
+ RES_UNITS_BYTES, mem_cgroup_read, mem_cgroup_write);
+ if (ret)
+ return ret;
+
    return cgroup_add_files(cont, ss, mem_cgroup_files,
        ARRAY_SIZE(mem_cgroup_files));
}

```

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

Subject: Re: [PATCH 2/3] Introduce the res_counter_populate() function
Posted by [kir](#) on Thu, 04 Oct 2007 12:11:57 GMT
[View Forum Message](#) <> [Reply to Message](#)

Pavel Emelyanov wrote:

```

> <...skipped...>
> +static char * units_names[RES_UNITS_MAX][RES_CFT_MAX] = {
> + [RES_UNITS_BYTES] = {
> + "usage_in_bytes",
> + "limit_in_bytes",
> + "failcnt",
> + },
> + [RES_UNITS_ITEMS] = {
> + "usage",
> + "limit",
> + "failcnt",
> + },
> +};
>

```

Sorry for being late in the game, but can we please bring back the issue of naming those files?

To me, names like "usage_in_bytes" doesn't really make much sense, unless we will also have something like "usage_in_pages" next to it -- i.e. use several different units for the same resource. I seriously

doubt we should...

Still, we need a way to denote units of measurement for each resource.
Here are the options I can think of:

0. Hardcode the name of measurement units in file name, as it is (partially -- only for "bytes" done now). Looks ugly to me, access interface will be inconsistent, not program-friendly: as file name is not "usage", but "usage*", it will be easy to implement in shell, but requires some additional logic in C.
1. One obvious way is to put it in documentation, i.e. say something like "this parameter is measured in bytes". The problem is documentation is a bit far away from the actual file we read the value from.
2. Put units into the file itself, i.e. "cat usage" will print something like "1024 bytes". This is very user-friendly, but not really program-friendly: while in C it is just `fscanf(fd, "%d", &val)`, shell users will require something like `cut -d ' ' -f1` to extract the numeric value.
3. Put units into a separate new files named "units" (or, well, "measurement_units" (or even "measured_in") if you are fan of long descriptive names). So, "cat units" will show us "bytes" or "items" or "pages"...
4. Encode units into a file name, like "measured_in_bytes". This would be a separate file which exist just for the sake of the name. This looks bad to be since getting this info from a program is complex.

I prefer approach #3 -- easy and consistent.

> <...skipped...>

>

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH 2/3] Introduce the `res_counter_populate()` function

Posted by [Paul Menage](#) on Thu, 04 Oct 2007 15:01:25 GMT

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

On 10/4/07, Kir Kolyshkin <kir@openvz.org> wrote:

>

- > 3. Put units into a separate new files named "units" (or, well,
> "measurement_units" (or even "measured_in") if you are fan of long

> descriptive names). So, "cat units" will show us "bytes" or "items" or
> "pages"...

I'd vote for this option (just "units"). Good idea. And I don't think
there's a need for any hard-coded enumerations - just make the creator
of the resource counter specify a const char* that can be returned in
the units file.

Paul

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

Subject: Re: [PATCH 1/3] Typedefs the read and write functions in cftype
Posted by [Paul Menage](#) on Thu, 04 Oct 2007 15:18:22 GMT
[View Forum Message](#) <> [Reply to Message](#)

On 10/4/07, Pavel Emelyanov <xemul@openvz.org> wrote:
> This is just to reduce the code amount in the future.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Acked-by: Paul Menage <menage@google.com>

>
> ---
>
> diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h
> index 8747932..0635004 100644
> --- a/include/linux/cgroup.h
> +++ b/include/linux/cgroup.h
> @@ -178,6 +178,15 @@ struct css_set {
> * - the 'cftype' of the file is file->f_dentry->d_fsdata
> */
>
> +struct cftype;
> +
> +typedef ssize_t (*cft_read) (struct cgroup *cont, struct cftype *cft,
> + struct file *file,
> + char __user *buf, size_t nbytes, loff_t *ppos);
> +typedef ssize_t (*cft_write) (struct cgroup *cont, struct cftype *cft,
> + struct file *file,
> + const char __user *buf, size_t nbytes, loff_t *ppos);
> +
> #define MAX_CFTYPE_NAME 64
> struct cftype {

```

> /* By convention, the name should begin with the name of the
> @@ -185,18 +194,14 @@ struct cftype {
> char name[MAX_CFTYPE_NAME];
> int private;
> int (*open) (struct inode *inode, struct file *file);
> - ssize_t (*read) (struct cgroup *cont, struct cftype *cft,
> - struct file *file,
> - char __user *buf, size_t nbytes, loff_t *ppos);
> + cft_read read;
> /*
> * read_uint() is a shortcut for the common case of returning a
> * single integer. Use it in place of read()
> */
> u64 (*read_uint) (struct cgroup *cont, struct cftype *cft);
> - ssize_t (*write) (struct cgroup *cont, struct cftype *cft,
> - struct file *file,
> - const char __user *buf, size_t nbytes, loff_t *ppos);
>
> + cft_write write;
> /*
> * write_uint() is a shortcut for the common case of accepting
> * a single integer (as parsed by simple_strtoll) from
>
>

```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH 2/3] Introduce the res_counter_populate() function

Posted by [Paul Menage](#) on Thu, 04 Oct 2007 15:22:36 GMT

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

On 10/4/07, Pavel Emelyanov <xemul@openvz.org> wrote:

```

> This one is responsible for initializing the RES_CFT_MAX files
> properly and register them inside the container.
>
> The caller must provide the cgroup, the cgroup_subsys, the
> RES_CFT_MAX * sizeof(cftype) chunk of zeroed memory, the
> units of measure and the read and write callbacks.

```

It would be really nice if we could avoid needing the caller to worry about the res_counter I/O. See the patch I sent last week entitled "simplify memory controller and resource counter I/O". It makes use of the read_unit() and write_unit() methods in res_counters, and combined with your patch hides all the I/O details from the subsystem.

Paul

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
