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