Subject: [PATCH 7/10] Bsdacct: stop using global bsd_acct_struct instance in internal functions
Posted by Pavel Emelianov on Thu, 10 Apr 2008 08:28:29 GMT
View Forum Message <> Reply to Message

This adds the appropriate pointer to all the internal (i.e. static)
functions that work with global acct instance. API calls pass this
global one to them by now.

Essentially this is a plain s/acct_globals./acct->/ over the file.

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

```
---
 kernel/acct.c |   77 +++++++++++++++++++++++++++++---------------------------
 1 files changed, 39 insertions(+), 38 deletions(-)

diff --git a/kernel/acct.c b/kernel/acct.c
index fc71c13..72d4760 100644
--- a/kernel/acct.c
+++ b/kernel/acct.c
@@ -75,7 +75,8 @@ int acct_parm[3] = {4, 2, 30};
 /*
  * External references and all of the globals.
  */
-static void do_acct_process(struct pid_namespace *ns, struct file *);
+static void do_acct_process(struct bsd_acct_struct *acct,
+  struct pid_namespace *ns, struct file *);

 /*
  * This structure is used so that all the data protected by lock
@@ -106,7 +107,7 @@ static void acct_timeout(unsigned long x)
 /*
  * Check the amount of free space and suspend/resume accordingly.
  */
-static int check_free_space(struct file *file)
+static int check_free_space(struct bsd_acct_struct *acct, struct file *file)
 {
  struct kstatfs sbuf;
  int res;
@@ -115,8 +116,8 @@ static int check_free_space(struct file *file)
  sector_t suspend;

  spin_lock(&acct_lock);
- res = acct_globals.active;
- if (!file || !acct_globals.needcheck)
+ res = acct->active;
+ if (!file || !acct->needcheck)
```

```
   goto out;
  spin_unlock(&acct_lock);

@@ -137,33 +138,33 @@ static int check_free_space(struct file *file)
   act = 0;

  /*
-  * If some joker switched acct_globals.file under us we'ld better be
+  * If some joker switched acct->file under us we'ld better be
   * silent and _not_ touch anything.
   */
  spin_lock(&acct_lock);
- if (file != acct_globals.file) {
+ if (file != acct->file) {
   if (act)
    res = act>0;
   goto out;
  }

- if (acct_globals.active) {
+ if (acct->active) {
   if (act < 0) {
-   acct_globals.active = 0;
+   acct->active = 0;
    printk(KERN_INFO "Process accounting paused\n");
   }
  } else {
   if (act > 0) {
-   acct_globals.active = 1;
+   acct->active = 1;
    printk(KERN_INFO "Process accounting resumed\n");
   }
  }

- del_timer(&acct_globals.timer);
- acct_globals.needcheck = 0;
- acct_globals.timer.expires = jiffies + ACCT_TIMEOUT*HZ;
- add_timer(&acct_globals.timer);
- res = acct_globals.active;
+ del_timer(&acct->timer);
+ acct->needcheck = 0;
+ acct->timer.expires = jiffies + ACCT_TIMEOUT*HZ;
+ add_timer(&acct->timer);
+ res = acct->active;
 out:
  spin_unlock(&acct_lock);
  return res;
@@ -175,34 +176,33 @@ out:
```

```
 *
 * NOTE: acct_lock MUST be held on entry and exit.
 */
-static void acct_file_reopen(struct file *file)
+static void acct_file_reopen(struct bsd_acct_struct *acct, struct file *file)
 {
  struct file *old_acct = NULL;
  struct pid_namespace *old_ns = NULL;

- if (acct_globals.file) {
-  old_acct = acct_globals.file;
-  old_ns = acct_globals.ns;
-  del_timer(&acct_globals.timer);
-  acct_globals.active = 0;
-  acct_globals.needcheck = 0;
-  acct_globals.file = NULL;
+ if (acct->file) {
+  old_acct = acct->file;
+  old_ns = acct->ns;
+  del_timer(&acct->timer);
+  acct->active = 0;
+  acct->needcheck = 0;
+  acct->file = NULL;
  }
  if (file) {
-  acct_globals.file = file;
-  acct_globals.ns = get_pid_ns(task_active_pid_ns(current));
-  acct_globals.needcheck = 0;
-  acct_globals.active = 1;
+  acct->file = file;
+  acct->ns = get_pid_ns(task_active_pid_ns(current));
+  acct->needcheck = 0;
+  acct->active = 1;
   /* It's been deleted if it was used before so this is safe */
-  setup_timer(&acct_globals.timer, acct_timeout,
-    (unsigned long)&acct_globals);
-  acct_globals.timer.expires = jiffies + ACCT_TIMEOUT*HZ;
-  add_timer(&acct_globals.timer);
+  setup_timer(&acct->timer, acct_timeout, (unsigned long)acct);
+  acct->timer.expires = jiffies + ACCT_TIMEOUT*HZ;
+  add_timer(&acct->timer);
  }
  if (old_acct) {
   mnt_unpin(old_acct->f_path.mnt);
   spin_unlock(&acct_lock);
-  do_acct_process(old_ns, old_acct);
+  do_acct_process(acct, old_ns, old_acct);
   filp_close(old_acct, NULL);
```

```
   put_pid_ns(old_ns);
   spin_lock(&acct_lock);
@@ -237,7 +237,7 @@ static int acct_on(char *name)

  spin_lock(&acct_lock);
  mnt_pin(file->f_path.mnt);
- acct_file_reopen(file);
+ acct_file_reopen(&acct_globals, file);
  spin_unlock(&acct_lock);

  mntput(file->f_path.mnt); /* it's pinned, now give up active reference */
@@ -273,7 +273,7 @@ asmlinkage long sys_acct(const char __user *name)
   error = security_acct(NULL);
   if (!error) {
    spin_lock(&acct_lock);
-   acct_file_reopen(NULL);
+   acct_file_reopen(&acct_globals, NULL);
    spin_unlock(&acct_lock);
   }
  }
@@ -291,7 +291,7 @@ void acct_auto_close_mnt(struct vfsmount *m)
 {
  spin_lock(&acct_lock);
  if (acct_globals.file && acct_globals.file->f_path.mnt == m)
-  acct_file_reopen(NULL);
+  acct_file_reopen(&acct_globals, NULL);
  spin_unlock(&acct_lock);
 }

@@ -307,7 +307,7 @@ void acct_auto_close(struct super_block *sb)
  spin_lock(&acct_lock);
  if (acct_globals.file &&
      acct_globals.file->f_path.mnt->mnt_sb == sb) {
-  acct_file_reopen(NULL);
+  acct_file_reopen(&acct_globals, NULL);
  }
  spin_unlock(&acct_lock);
 }
@@ -426,7 +426,8 @@ static u32 encode_float(u64 value)
 /*
  *  do_acct_process does all actual work. Caller holds the reference to file.
  */
-static void do_acct_process(struct pid_namespace *ns, struct file *file)
+static void do_acct_process(struct bsd_acct_struct *acct,
+  struct pid_namespace *ns, struct file *file)
 {
  struct pacct_struct *pacct = &current->signal->pacct;
  acct_t ac;
```

```
@@ -441,7 +442,7 @@ static void do_acct_process(struct pid_namespace *ns, struct file *file)
   * First check to see if there is enough free_space to continue
   * the process accounting system.
   */
- if (!check_free_space(file))
+ if (!check_free_space(acct, file))
   return;

  /*
@@ -604,7 +605,7 @@ void acct_process(void)
  ns = get_pid_ns(acct_globals.ns);
  spin_unlock(&acct_lock);

- do_acct_process(ns, file);
+ do_acct_process(&acct_globals, ns, file);
  fput(file);
  put_pid_ns(ns);
 }
--
1.5.3.4
```

_____

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