Subject: [RFC 2/7] consolidate all res_counter manipulation
Posted by Glauber Costa on Fri, 30 Mar 2012 08:04:40 GMT
View Forum Message <> Reply to Message

This patch moves all the locked updates done to res_counter to
__res_counter_add(). It gets flags for the special cases like nofail(),
and a negative value of the increment means uncharge.

This will be useful later when we start doing percpu_counter updates.

Signed-off-by: Glauber Costa <glommer@parallels.com>

```
---
 kernel/res_counter.c |   59 +++++++++++++++++++++++++++++++++------------------------------
 1 files changed, 28 insertions(+), 31 deletions(-)

diff --git a/kernel/res_counter.c b/kernel/res_counter.c
index b8a3d6a..ecb4aad 100644
--- a/kernel/res_counter.c
+++ b/kernel/res_counter.c
@@ -22,17 +22,32 @@ void res_counter_init(struct res_counter *counter, struct res_counter *parent)
  counter->parent = parent;
 }

-int res_counter_charge_locked(struct res_counter *counter, unsigned long val)
+int __res_counter_add(struct res_counter *c, long val, bool fail)
 {
- if (counter->usage + val > counter->limit) {
-  counter->failcnt++;
-  return -ENOMEM;
+ int ret = 0;
+ u64 usage;
+
+ spin_lock(&c->lock);
+
+ usage = c->usage;
+
+ if (usage + val > c->limit) {
+  c->failcnt++;
+  ret = -ENOMEM;
+  if (fail)
+   goto out;
  }

- counter->usage += val;
- if (counter->usage > counter->max_usage)
-  counter->max_usage = counter->usage;
- return 0;
```

```
+ usage += val;
+
+ c->usage = usage;
+ if (usage > c->max_usage)
+  c->max_usage = usage;
+
+out:
+ spin_unlock(&c->lock);
+ return ret;
+
 }

 int res_counter_charge(struct res_counter *counter, unsigned long val,
@@ -45,9 +60,7 @@ int res_counter_charge(struct res_counter *counter, unsigned long val,
  *limit_fail_at = NULL;
  local_irq_save(flags);
  for (c = counter; c != NULL; c = c->parent) {
-  spin_lock(&c->lock);
-  ret = res_counter_charge_locked(c, val);
-  spin_unlock(&c->lock);
+  ret = __res_counter_add(c, val, true);
   if (ret < 0) {
    *limit_fail_at = c;
    goto undo;
@@ -57,9 +70,7 @@ int res_counter_charge(struct res_counter *counter, unsigned long val,
  goto done;
 undo:
  for (u = counter; u != c; u = u->parent) {
-  spin_lock(&u->lock);
-  res_counter_uncharge_locked(u, val);
-  spin_unlock(&u->lock);
+  __res_counter_add(u, -val, false);
  }
 done:
  local_irq_restore(flags);
@@ -77,11 +88,7 @@ int res_counter_charge_nofail(struct res_counter *counter, unsigned long
val,
  *limit_fail_at = NULL;
  local_irq_save(flags);
  for (c = counter; c != NULL; c = c->parent) {
-  spin_lock(&c->lock);
-  r = res_counter_charge_locked(c, val);
-  if (r)
-   c->usage += val;
-  spin_unlock(&c->lock);
+  r = __res_counter_add(c, val, false);
   if (r < 0 && ret == 0) {
    *limit_fail_at = c;
```

```
    ret = r;
@@ -91,13 +98,6 @@ int res_counter_charge_nofail(struct res_counter *counter, unsigned long
val,

  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)
 {
@@ -105,11 +105,8 @@ void res_counter_uncharge(struct res_counter *counter, unsigned long
val)
  struct res_counter *c;

  local_irq_save(flags);
- for (c = counter; c != NULL; c = c->parent) {
-  spin_lock(&c->lock);
-  res_counter_uncharge_locked(c, val);
-  spin_unlock(&c->lock);
- }
+ for (c = counter; c != NULL; c = c->parent)
+  __res_counter_add(c, -val, false);
  local_irq_restore(flags);
 }

--
1.7.4.1
```