Subject: Re: [RFC][PATCH] allow "unlimited" limit value.
Posted by David Rientjes on Tue, 25 Sep 2007 19:07:58 GMT
View Forum Message <> Reply to Message

On Wed, 26 Sep 2007, KAMEZAWA Hiroyuki wrote:

> > > #define RES_COUNTER_INIFINITY   (~0ULL)
> > > or some nice name
> >
> > Why do we need this at all? We can simply push -1 there and be happy.
> >
> Hm, can this work now ?
> ==
> echo -1 > /cgroup/memory.limit_in_bytes
> ==
> Or users have to do following for unlimit resource ?
> ==
> echo some-very-very-big-number > /cgroup/memory.limit_in_bytes
>
>
> I just think when some special value "-1" has a nice nick name, users will
> be happy. If I'm a novice user, I don't imagine I can write -1 to limit value.
> (but ok, tools can hide it for them.)
>

Please simply use 0 to denote unconstrained memory, it's quite obvious
that nobody will sanely attach tasks to a cgroup that has no bytes of
memory allowed.

In fact, I proposed this in a patch on August 27.

I really don't like the use of ULONG_MAX to denote the absence of any
memory controls for a particular container. I think 0 would be suitable
since its use doesn't make any logical sense (you're not going to be
assigning a set of tasks to a resource void of pages).

Signed-off-by: David Rientjes <rientjes@google.com>

```
---
 Documentation/controllers/memory.txt |    5 ++++-
 kernel/res_counter.c                 |    7 +++++--
 2 files changed, 9 insertions(+), 3 deletions(-)

diff --git a/Documentation/controllers/memory.txt b/Documentation/controllers/memory.txt
--- a/Documentation/controllers/memory.txt
+++ b/Documentation/controllers/memory.txt
@@ -164,13 +164,16 @@ c. Enable CONFIG_CONTAINER_MEM_CONT
```

```
 # echo $$ >  /containers/0/tasks

 Since now we're in the 0 container,
-We can alter the memory limit:
+We can alter the memory limit (in pages):
 # echo -n 6000 > /containers/0/memory.limit

 We can check the usage:
 # cat /containers/0/memory.usage
 25

+If memory.limit is set to 0, no charge is accumlated for that resource
+controller.
+
 The memory.failcnt field gives the number of times that the container limit was
 exceeded.

diff --git a/kernel/res_counter.c b/kernel/res_counter.c
--- a/kernel/res_counter.c
+++ b/kernel/res_counter.c
@@ -16,12 +16,15 @@
 void res_counter_init(struct res_counter *counter)
 {
  spin_lock_init(&counter->lock);
- counter->limit = (unsigned long)LONG_MAX;
 }

 int res_counter_charge_locked(struct res_counter *counter, unsigned long val)
 {
- if (counter->usage + val > counter->limit) {
+ /*
+  * If 'memory.limit' is set to 0, there is no charge to this
+  * res_counter.
+  */
+ if (counter->limit && counter->usage + val > counter->limit) {
   counter->failcnt++;
   return -ENOMEM;
  }
```

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