Subject: [PATCH 0/4] Container Freezer: Reuse Suspend Freezer
Posted by Matt Helsley on Mon, 07 Jul 2008 22:58:23 GMT
View Forum Message <> Reply to Message

This patchset reuses the container infrastructure and the swsusp freezer to
freeze a group of tasks.

The freezer subsystem in the container filesystem defines a file named
freezer.state. Writing "FROZEN" to the state file will freeze all tasks in the
cgroup. Subsequently writing "RUNNING" will unfreeze the tasks in the cgroup.
Reading will return the current state.

* Examples of usage :

```
  # mkdir /containers/freezer
  # mount -t cgroup -ofreezer,signal freezer  /containers
  # mkdir /containers/0
  # echo $some_pid > /containers/0/tasks
```

to get status of the freezer subsystem :

```
  # cat /containers/0/freezer.state
  RUNNING
```

to freeze all tasks in the container :

```
  # echo FROZEN > /containers/0/freezer.state
  # cat /containers/0/freezer.state
  FREEZING
  # cat /containers/0/freezer.state
  FROZEN
```

to unfreeze all tasks in the container :

```
  # echo RUNNING > /containers/0/freezer.state
  # cat /containers/0/freezer.state
  RUNNING
```

to kill all tasks in the container :

```
  # echo 9 > /containers/0/signal.kill
```

I've reworked Cedric's patches to use task_lock() to protect access to the
task's cgroup.

Paul, Pavel asked me to send these to Rafael next. They are patches to make
the freezer useful for checkpoint/restart using cgroups so it would be nice
to get an explicit [N]Ack from you first.

Rafael, if Paul agrees, please consider applying these patches.

Changes since v3:
v4 (Almost all of these changes are confined to patch 3):
 Reworked the series to use task_lock() instead of RCU.
 Reworked the series to use write_string() and read_seq_string()
  cgroup methods.
 Fixed the race Paul Menage identified.
 Fixed up check_if_frozen() to do more than just test the FROZEN
  flag. In some cases tasks could be stopped (T) and marked
  FREEZING. When that happens we can safely assume that it
  will be frozen immediately upon waking up in the kernel.
  Waiting for it to get marked with PF_FROZEN in order to
  transition to the FROZEN state would block unnecessarily.
 Removed freezer_ prefix from static functions in cgroup_freezer.c.
 Simplified STATE_ switch.
 Updated the locking comments.
v3:
 Ported to 2.6.26-rc5-mm2 with Rafael's freezer patches
 Tested on 24 combinations of 3 architectures (x86, x86_64, ppc64)
  with 8 different kernel configs varying power management
  and cgroup config variables. Each patch builds and boots
  in these 24 combinations.
 Passes functional testing.
v2 (roughly patches 3 and 5):
 Moved the "kill" file into a separate cgroup subsystem (signal) and
  it's own patch.
 Changed the name of the file from freezer.freeze to freezer.state.
 Switched from taking 1 and 0 as input to the strings "FROZEN" and
  "RUNNING", respectively. This helps keep the interface
  human-usable if/when we need to more states.
 Checked that stopped or interrupted is "frozen enough"
  Since try_to_freeze() is called upon wakeup of these tasks
  this should be fine. This idea comes from recent changes to
  the freezer.
 Checked that if (task == current) whilst freezing cgroup we're ok
 Fixed bug where -EBUSY would always be returned when freezing
 Added code to handle userspace retries for any remaining -EBUSY

Cheers,
 -Matt Helsley

--