Subject: [RFC][PATCH 4/4] container freezer: do not unfreeze a frozen container when the system is resumed
Posted by Cedric Le Goater on Wed, 20 Jun 2007 15:08:03 GMT
View Forum Message <> Reply to Message

From: Cedric Le Goater <clg@fr.ibm.com>

When a system is resumed after a suspend, it will also unfreeze
frozen containers.

This patchs modifies the resume sequence to skip the tasks which
are part of a frozen container.

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
---
 include/linux/container_freezer.h | 56 ++
 kernel/container_freezer.c | 20 +----------
 kernel/power/process.c | 4 ++
 3 files changed, 62 insertions(+), 18 deletions(-)

Index: 2.6.22-rc4-mm2/include/linux/container_freezer.h
===
--- /dev/null
+++ 2.6.22-rc4-mm2/include/linux/container_freezer.h
@@ -0,0 +1,56 @@
+#ifndef _LINUX_CONTAINER_FREEZER_H
+#define _LINUX_CONTAINER_FREEZER_H
+/*
+ * container_freezer.h - container freezer subsystem interface
+ *
+ * Copyright (C) Cedric Le Goater, IBM Corp. 2007
+ *
+ */
+
+#include <linux/container.h>
+
+#ifdef CONFIG_CONTAINER_FREEZER
+
+enum freezer_state {
+ STATE_RUNNING = 0,
+ STATE_FREEZING,
+ STATE_FROZEN,
+};
+
+struct freezer {
+ struct container_subsys_state css;
+ enum freezer_state state;
+ spinlock_t lock;

```
+};
+
+static inline struct freezer *container_freezer(
+  struct container *container)
+{
+ return container_of(
+  container_subsys_state(container, freezer_subsys_id),
+  struct freezer, css);
+}
+
+static inline int container_frozen(struct task_struct *task)
+{
+ struct container *container = task_container(task, freezer_subsys_id);
+ struct freezer *freezer = container_freezer(container);
+ enum freezer_state state;
+
+ spin_lock(&freezer->lock);
+ state = freezer->state;
+ spin_unlock(&freezer->lock);
+
+ return (state == STATE_FROZEN);
+}
+
+#else /* !CONFIG_CONTAINER_FREEZER */
+
+static inline int container_frozen(struct task_struct *task)
+{
+ return 0;
+}
+
+#endif /* !CONFIG_CONTAINER_FREEZER */
+
+#endif /* _LINUX_CONTAINER_FREEZER_H */
Index: 2.6.22-rc4-mm2/kernel/container_freezer.c
===================================================================
--- 2.6.22-rc4-mm2.orig/kernel/container_freezer.c
+++ 2.6.22-rc4-mm2/kernel/container_freezer.c
@@ -9,12 +9,8 @@
 #include <linux/fs.h>
 #include <linux/uaccess.h>
 #include <linux/freezer.h>
-
-enum freezer_state {
- STATE_RUNNING = 0,
- STATE_FREEZING,
- STATE_FROZEN,
-};
+#include <linux/container.h>
```

```
+#include <linux/container_freezer.h>

 static const char *freezer_state_strs[] = {
  "RUNNING\n",
@@ -22,21 +18,9 @@ static const char *freezer_state_strs[]
  "FROZEN\n"
 };

-struct freezer {
- struct container_subsys_state css;
- enum freezer_state state;
- spinlock_t lock;
-};

 struct container_subsys freezer_subsys;

-static inline struct freezer *container_freezer(
-  struct container *container)
-{
- return container_of(
-  container_subsys_state(container, freezer_subsys_id),
-  struct freezer, css);
-}

 static int freezer_create(struct container_subsys *ss,
       struct container *container)
Index: 2.6.22-rc4-mm2/kernel/power/process.c
===================================================================
--- 2.6.22-rc4-mm2.orig/kernel/power/process.c
+++ 2.6.22-rc4-mm2/kernel/power/process.c
@@ -13,6 +13,7 @@
 #include <linux/module.h>
 #include <linux/syscalls.h>
 #include <linux/freezer.h>
+#include <linux/container_freezer.h>

 /*
  * Timeout for stopping processes
@@ -169,6 +170,9 @@ static void thaw_tasks(int thaw_user_spa
    if (!p->mm == thaw_user_space)
     continue;

+  if (container_frozen(p))
+   continue;
+
    thaw_process(p);
   } while_each_thread(g, p);
   read_unlock(&tasklist_lock);
```

--

_____

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