
Subject: [RFC][PATCH 2/5] Container Freezer: Make refrigerator always available
Posted by [Matt Helsley](#) on Thu, 24 Apr 2008 06:47:58 GMT
[View Forum Message](#) <> [Reply to Message](#)

Now that the TIF_FREEZE flag is available in all architectures,
extract the refrigerator() and freeze_task() from kernel/power/process.c
and make it available to all.

The refrigerator() can now be used in a control group subsystem
implementing a control group freezer.

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Signed-off-by: Matt Helsley <matthltc@us.ibm.com>
Tested-by: Matt Helsley <matthltc@us.ibm.com>

Changelog:

Merged Roland's "STOPPED is frozen enough" changes. For details see:
<http://lkml.org/lkml/2008/3/3/676>

```
include/linux/freezer.h | 19 ++-----
kernel/Makefile         |  4 -
kernel/freezer.c        | 124 +++++++++++++++++++++++++++++++++++++++++++++++++++++
kernel/power/process.c  | 113 -----
4 files changed, 133 insertions(+), 127 deletions(-)
```

Index: linux-2.6.25-mm1/include/linux/freezer.h

=====

--- linux-2.6.25-mm1.orig/include/linux/freezer.h

+++ linux-2.6.25-mm1/include/linux/freezer.h

@@ -4,11 +4,10 @@

#define FREEZER_H_INCLUDED

#include <linux/sched.h>

#include <linux/wait.h>

-#ifdef CONFIG_PM_SLEEP

/*

* Check if a process has been frozen

*/

static inline int frozen(struct task_struct *p)

{

@@ -61,22 +60,27 @@ static inline int thaw_process(struct ta

task_unlock(p);

return 0;

}

extern void refrigerator(void);

-extern int freeze_processes(void);

```

-extern void thaw_processes(void);

static inline int try_to_freeze(void)
{
    if (freezing(current)) {
        refrigerator();
        return 1;
    } else
        return 0;
}

+extern int freeze_task(struct task_struct *p, int with_mm_only);
+
+
+
+
+extern int freeze_processes(void);
+extern void thaw_processes(void);
+
+
+/*
+ * The PF_FREEZER_SKIP flag should be set by a vfork parent right before it
+ * calls wait_for_completion(&vfork) and reset right after it returns from this
+ * function. Next, the parent should call try_to_freeze() to freeze itself
+ * appropriately in case the child has exited before the freezing of tasks is
@@ -156,22 +160,13 @@ static inline void set_freezable(void)
    __retval); \
} while (try_to_freeze()); \
__retval; \
})
#else /* !CONFIG_PM_SLEEP */
-static inline int frozen(struct task_struct *p) { return 0; }
-static inline int freezing(struct task_struct *p) { return 0; }
-static inline void set_freeze_flag(struct task_struct *p) {}
-static inline void clear_freeze_flag(struct task_struct *p) {}
-static inline int thaw_process(struct task_struct *p) { return 1; }
-
-static inline void refrigerator(void) {}
static inline int freeze_processes(void) { BUG(); return 0; }
static inline void thaw_processes(void) {}

-static inline int try_to_freeze(void) { return 0; }
-
static inline void freezer_do_not_count(void) {}
static inline void freezer_count(void) {}
static inline int freezer_should_skip(struct task_struct *p) { return 0; }
static inline void set_freezable(void) {}

```

Index: linux-2.6.25-mm1/kernel/Makefile

=====

```

--- linux-2.6.25-mm1.orig/kernel/Makefile
+++ linux-2.6.25-mm1/kernel/Makefile
@@ -3,15 +3,15 @@
#

obj-y    = sched.o fork.o exec_domain.o panic.o printk.o profile.o \
    exit.o itimer.o time.o softirq.o resource.o \
    sysctl.o capability.o ptrace.o timer.o user.o \
-   signal.o sys.o kmod.o workqueue.o pid.o \
+   signal.o sys.o kmod.o workqueue.o pid.o freezer.o \
    rcupdate.o extable.o params.o posix-timers.o \
    kthread.o wait.o kfifo.o sys_ni.o posix-cpu-timers.o mutex.o \
    hrtimer.o rwsem.o nsproxy.o srcu.o semaphore.o \
-   notifier.o ksysfs.o pm_qos_params.o
+   notifier.o ksysfs.o pm_qos_params.o

ifdef CONFIG_FTRACE
# Do not profile debug utilities
ORIG_CFLAGS := $(KBUILD_CFLAGS)
KBUILD_CFLAGS = $(if $(filter-out lockdep% %debug,$(basename $(notdir $@))), \
Index: linux-2.6.25-mm1/kernel/freezer.c
=====
--- /dev/null
+++ linux-2.6.25-mm1/kernel/freezer.c
@@ -0,0 +1,124 @@
+/*
+ * kernel/freezer.c - Function to freeze a process
+ *
+ * Originally from kernel/power/process.c
+ */
+
+#include <linux/interrupt.h>
+#include <linux/suspend.h>
+#include <linux/module.h>
+#include <linux/syscalls.h>
+#include <linux/freezer.h>
+
+/*
+ * freezing is complete, mark current process as frozen
+ */
+static inline void frozen_process(void)
+{
+ if (!unlikely(current->flags & PF_NOFREEZE)) {
+  current->flags |= PF_FROZEN;
+  wmb();
+ }
+ clear_freeze_flag(current);
+}

```

```

+
+/* Refrigerator is place where frozen processes are stored :-). */
+void refrigerator(void)
+{
+ /* Hmm, should we be allowed to suspend when there are realtime
+  processes around? */
+ long save;
+
+ task_lock(current);
+ if (freezing(current)) {
+  frozen_process();
+  task_unlock(current);
+ } else {
+  task_unlock(current);
+  return;
+ }
+ save = current->state;
+ pr_debug("%s entered refrigerator\n", current->comm);
+
+ spin_lock_irq(&current->sighand->siglock);
+ recalc_sigpending(); /* We sent fake signal, clean it up */
+ spin_unlock_irq(&current->sighand->siglock);
+
+ for (;;) {
+  set_current_state(TASK_UNINTERRUPTIBLE);
+  if (!frozen(current))
+   break;
+  schedule();
+ }
+ pr_debug("%s left refrigerator\n", current->comm);
+ __set_current_state(save);
+}
+EXPORT_SYMBOL(refrigerator);
+
+static void fake_signal_wake_up(struct task_struct *p)
+{
+ unsigned long flags;
+
+ spin_lock_irqsave(&p->sighand->siglock, flags);
+ signal_wake_up(p, 0);
+ spin_unlock_irqrestore(&p->sighand->siglock, flags);
+}
+
+static int has_mm(struct task_struct *p)
+{
+ return (p->mm && !(p->flags & PF_BORROWED_MM));
+}
+

```

```

+/**
+ * freeze_task - send a freeze request to given task
+ * @p: task to send the request to
+ * @with_mm_only: if set, the request will only be sent if the task has its
+ * own mm
+ * Return value: 0, if @with_mm_only is set and the task has no mm of its
+ * own or the task is frozen, 1, otherwise
+ *
+ * The freeze request is sent by setting the task's TIF_FREEZE flag and
+ * either sending a fake signal to it or waking it up, depending on whether
+ * or not it has its own mm (ie. it is a user land task). If @with_mm_only
+ * is set and the task has no mm of its own (ie. it is a kernel thread),
+ * its TIF_FREEZE flag should not be set.
+ *
+ * The task_lock() is necessary to prevent races with exit_mm() or
+ * use_mm()/unuse_mm() from occurring.
+ */
+int freeze_task(struct task_struct *p, int with_mm_only)
+{
+ int ret = 1;
+
+ task_lock(p);
+ if (freezing(p)) {
+ if (has_mm(p)) {
+ if (!signal_pending(p))
+ fake_signal_wake_up(p);
+ } else {
+ if (with_mm_only)
+ ret = 0;
+ else
+ wake_up_state(p, TASK_INTERRUPTIBLE);
+ }
+ } else {
+ rmb();
+ if (frozen(p)) {
+ ret = 0;
+ } else {
+ if (has_mm(p)) {
+ set_freeze_flag(p);
+ fake_signal_wake_up(p);
+ } else {
+ if (with_mm_only) {
+ ret = 0;
+ } else {
+ set_freeze_flag(p);
+ wake_up_state(p, TASK_INTERRUPTIBLE);
+ }
+ }
+ }
+ }
+ }

```

```

+ }
+ }
+ task_unlock(p);
+ return ret;
+}
Index: linux-2.6.25-mm1/kernel/power/process.c
=====
--- linux-2.6.25-mm1.orig/kernel/power/process.c
+++ linux-2.6.25-mm1/kernel/power/process.c
@@ -29,121 +29,10 @@ static inline int freezeable(struct task
    (p->exit_state != 0))
    return 0;
    return 1;
}

-/*
- * freezing is complete, mark current process as frozen
- */
-static inline void frozen_process(void)
-{
- if (!unlikely(current->flags & PF_NOFREEZE)) {
-   current->flags |= PF_FROZEN;
-   wmb();
- }
- clear_freeze_flag(current);
-}
-
-/* Refrigerator is place where frozen processes are stored :-). */
-void refrigerator(void)
-{
- /* Hmm, should we be allowed to suspend when there are realtime
-  processes around? */
- long save;
-
- task_lock(current);
- if (freezing(current)) {
-   frozen_process();
-   task_unlock(current);
- } else {
-   task_unlock(current);
-   return;
- }
- save = current->state;
- pr_debug("%s entered refrigerator\n", current->comm);
-
- spin_lock_irq(&current->sighand->siglock);
- recalc_sigpending(); /* We sent fake signal, clean it up */
- spin_unlock_irq(&current->sighand->siglock);

```

```

-
- for (;;) {
-     set_current_state(TASK_UNINTERRUPTIBLE);
-     if (!frozen(current))
-         break;
-     schedule();
- }
- pr_debug("%s left refrigerator\n", current->comm);
- __set_current_state(save);
-}
-
-static void fake_signal_wake_up(struct task_struct *p)
-{
- unsigned long flags;
-
- spin_lock_irqsave(&p->sigband->siglock, flags);
- signal_wake_up(p, 0);
- spin_unlock_irqrestore(&p->sigband->siglock, flags);
-}
-
-static int has_mm(struct task_struct *p)
-{
- return (p->mm && !(p->flags & PF_BORROWED_MM));
-}
-
-/**
- * freeze_task - send a freeze request to given task
- * @p: task to send the request to
- * @with_mm_only: if set, the request will only be sent if the task has its
- * own mm
- * Return value: 0, if @with_mm_only is set and the task has no mm of its
- * own or the task is frozen, 1, otherwise
- *
- * The freeze request is sent by setting the task's TIF_FREEZE flag and
- * either sending a fake signal to it or waking it up, depending on whether
- * or not it has its own mm (ie. it is a user land task). If @with_mm_only
- * is set and the task has no mm of its own (ie. it is a kernel thread),
- * its TIF_FREEZE flag should not be set.
- *
- * The task_lock() is necessary to prevent races with exit_mm() or
- * use_mm()/unuse_mm() from occurring.
- */
-static int freeze_task(struct task_struct *p, int with_mm_only)
-{
- int ret = 1;
-
- task_lock(p);
- if (freezing(p)) {

```

```

- if (has_mm(p)) {
- if (!signal_pending(p))
- fake_signal_wake_up(p);
- } else {
- if (with_mm_only)
- ret = 0;
- else
- wake_up_state(p, TASK_INTERRUPTIBLE);
- }
- } else {
- rmb();
- if (frozen(p)) {
- ret = 0;
- } else {
- if (has_mm(p)) {
- set_freeze_flag(p);
- fake_signal_wake_up(p);
- } else {
- if (with_mm_only) {
- ret = 0;
- } else {
- set_freeze_flag(p);
- wake_up_state(p, TASK_INTERRUPTIBLE);
- }
- }
- }
- }
- task_unlock(p);
- return ret;
-}

static void cancel_freezing(struct task_struct *p)
{
    unsigned long flags;

@@ -274,7 +163,5 @@ void thaw_processes(void)
    thaw_tasks(FREEZER_KERNEL_THREADS);
    thaw_tasks(FREEZER_USER_SPACE);
    schedule();
    printk("done.\n");
}
-
-EXPORT_SYMBOL(refrigerator);

--

```

Containers mailing list
Containers@lists.linux-foundation.org

