Subject: Re: [PATCH -mm] Hook up group scheduler with control groups
Posted by Dhaval Giani on Thu, 27 Sep 2007 18:04:15 GMT
View Forum Message <> Reply to Message

On Thu, Sep 27, 2007 at 10:40:05AM -0700, Randy Dunlap wrote:
> On Thu, 27 Sep 2007 20:36:24 +0530 Srivatsa Vaddagiri wrote:
> Hi :)
>
> Uh, a few of my previous comments weren't fixed... (below)
>
>
> > --
> >
> > Enable "cgroup" (formerly containers) based fair group scheduling.
> > This will let administrator create arbitrary groups of tasks (using
> > "cgroup" pseudo filesystem) and control their cpu bandwidth usage.
> >
> > Signed-off-by : Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com>
> > Signed-off-by : Dhaval Giani <dhaval@linux.vnet.ibm.com>
> >
> > ---
> > include/linux/cgroup_subsys.h | 6 ++
> > init/Kconfig | 24 +++++---
> > kernel/sched.c | 122 ++
> > 3 files changed, 145 insertions(+), 7 deletions(-)
> >
>
> > Index: current/init/Kconfig
> > ===
> > --- current.orig/init/Kconfig
> > +++ current/init/Kconfig
> > @@ -319,6 +319,13 @@ config CPUSETS
> >
> > Say N if unsure.
> >
> > +config RESOURCE_COUNTERS
> > + bool "Resource counters"
> > + help
> > + This option enables controller independent resource accounting
> > + infrastructure that works with cgroups
>
> Use tab + 2 spaces consistently for help text indentation.
> End that sentence with a ".".
>

Hi,

hmmmm. So I looked into the sources with the patches applied, and they

don't have an indentation problem. It looks fine. I'm not sure why the
patch is getting generated like this though. Will fix up the '.' though.

Here it is,

Thanks,
--

Signed-off-by : Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com>
Signed-off-by : Dhaval Giani <dhaval@linux.vnet.ibm.com>

---
 include/linux/cgroup_subsys.h | 6 ++
 init/Kconfig | 24 +++++---
 kernel/sched.c | 122 ++
 3 files changed, 145 insertions(+), 7 deletions(-)

Index: current/include/linux/cgroup_subsys.h
===
--- current.orig/include/linux/cgroup_subsys.h
+++ current/include/linux/cgroup_subsys.h
@@ -36,3 +36,9 @@ SUBSYS(mem_cgroup)
 #endif

 /* */
+
+#ifdef CONFIG_FAIR_CGROUP_SCHED
+SUBSYS(cpu_cgroup)
+#endif
+
+/* */
Index: current/init/Kconfig
===
--- current.orig/init/Kconfig
+++ current/init/Kconfig
@@ -319,6 +319,13 @@ config CPUSETS

    Say N if unsure.

+config RESOURCE_COUNTERS
+ bool "Resource counters"
+ help
+   This option enables controller independent resource accounting
+   infrastructure that works with cgroups.
+ depends on CGROUPS
+
 config FAIR_GROUP_SCHED
  bool "Fair group CPU scheduler"

```
  default y
@@ -338,14 +345,17 @@ config FAIR_USER_SCHED
    This option will choose userid as the basis for grouping
    tasks, thus providing equal CPU bandwidth to each user.

-endchoice
+config FAIR_CGROUP_SCHED
+ bool "Control groups"
+  depends on CGROUPS
+  help
+   This option allows you to create arbitrary task groups
+   using the "cgroup" pseudo filesystem and control
+   the cpu bandwidth allocated to each such task group.
+   Refer to Documentation/cgroups.txt for more information
+   on "cgroup" pseudo filesystem.

-config RESOURCE_COUNTERS
- bool "Resource counters"
- help
-   This option enables controller independent resource accounting
-         infrastructure that works with cgroups
- depends on CGROUPS
+endchoice

 config SYSFS_DEPRECATED
  bool "Create deprecated sysfs files"
Index: current/kernel/sched.c
===================================================================
--- current.orig/kernel/sched.c
+++ current/kernel/sched.c
@@ -177,10 +177,16 @@ struct rt_prio_array {

 #ifdef CONFIG_FAIR_GROUP_SCHED

+#include <linux/cgroup.h>
+
 struct cfs_rq;

 /* task group related information */
 struct task_grp {
+#ifdef CONFIG_FAIR_CGROUP_SCHED
+ struct cgroup_subsys_state css;
+#endif
+
  /* schedulable entities of this group on each cpu */
  struct sched_entity **se;
  /* runqueue "owned" by this group on each cpu */
@@ -219,6 +225,9 @@ static inline struct task_grp *task_grp(
```

```
 #ifdef CONFIG_FAIR_USER_SCHED
  tg = p->user->tg;
+#elif CONFIG_FAIR_CGROUP_SCHED
+ tg = container_of(task_subsys_state(p, cpu_cgroup_subsys_id),
+     struct task_grp, css);
 #else
  tg  = &init_task_grp;
 #endif
@@ -6958,3 +6967,116 @@ int sched_group_set_shares(struct task_g
 }

 #endif  /* CONFIG_FAIR_GROUP_SCHED */
+
+#ifdef CONFIG_FAIR_CGROUP_SCHED
+
+/* return corresponding task_grp object of a cgroup */
+static inline struct task_grp *cgroup_tg(struct cgroup *cont)
+{
+ return container_of(cgroup_subsys_state(cont, cpu_cgroup_subsys_id),
+       struct task_grp, css);
+}
+
+static struct cgroup_subsys_state *
+cpu_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
+{
+ struct task_grp *tg;
+
+ if (!cont->parent) {
+  /* This is early initialization for the top cgroup */
+  init_task_grp.css.cgroup = cont;
+  return &init_task_grp.css;
+ }
+
+ /* we support only 1-level deep hierarchical scheduler atm */
+ if (cont->parent->parent)
+  return ERR_PTR(-EINVAL);
+
+ tg = sched_create_group();
+ if (IS_ERR(tg))
+  return ERR_PTR(-ENOMEM);
+
+ /* Bind the cgroup to task_grp object we just created */
+ tg->css.cgroup = cont;
+
+ return &tg->css;
+}
+
```

```
+static void cpu_cgroup_destroy(struct cgroup_subsys *ss,
+      struct cgroup *cont)
+{
+ struct task_grp *tg = cgroup_tg(cont);
+
+ sched_destroy_group(tg);
+}
+
+static int cpu_cgroup_can_attach(struct cgroup_subsys *ss,
+         struct cgroup *cont, struct task_struct *tsk)
+{
+ /* We don't support RT-tasks being in separate groups */
+ if (tsk->sched_class != &fair_sched_class)
+  return -EINVAL;
+
+ return 0;
+}
+
+static void
+cpu_cgroup_attach(struct cgroup_subsys *ss, struct cgroup *cont,
+    struct cgroup *old_cont, struct task_struct *tsk)
+{
+ sched_move_task(tsk);
+}
+
+static ssize_t cpu_shares_write(struct cgroup *cont, struct cftype *cftype,
+     struct file *file, const char __user *userbuf,
+     size_t nbytes, loff_t *ppos)
+{
+ unsigned long shareval;
+ struct task_grp *tg = cgroup_tg(cont);
+ char buffer[2*sizeof(unsigned long) + 1];
+ int rc;
+
+ if (nbytes > 2*sizeof(unsigned long)) /* safety check */
+  return -E2BIG;
+
+ if (copy_from_user(buffer, userbuf, nbytes))
+  return -EFAULT;
+
+ buffer[nbytes] = 0; /* nul-terminate */
+ shareval = simple_strtoul(buffer, NULL, 10);
+
+ rc = sched_group_set_shares(tg, shareval);
+
+ return (rc < 0 ? rc : nbytes);
+}
+
```

```
+static u64 cpu_shares_read_uint(struct cgroup *cont, struct cftype *cft)
+{
+ struct task_grp *tg = cgroup_tg(cont);
+
+ return (u64) tg->shares;
+}
+
+static struct cftype cpu_shares = {
+ .name = "shares",
+ .read_uint = cpu_shares_read_uint,
+ .write = cpu_shares_write,
+};
+
+static int cpu_cgroup_populate(struct cgroup_subsys *ss, struct cgroup *cont)
+{
+ return cgroup_add_file(cont, ss, &cpu_shares);
+}
+
+struct cgroup_subsys cpu_cgroup_subsys = {
+ .name       = "cpu",
+ .create     = cpu_cgroup_create,
+ .destroy    = cpu_cgroup_destroy,
+ .can_attach  = cpu_cgroup_can_attach,
+ .attach     = cpu_cgroup_attach,
+ .populate   = cpu_cgroup_populate,
+ .subsys_id   = cpu_cgroup_subsys_id,
+ .early_init = 1,
+};
+
+#endif /* CONFIG_FAIR_CGROUP_SCHED */
--
regards,
Dhaval
```