Subject: [PATCH] cgroup: support checking of subsystem dependencies
Posted by Li Zefan on Wed, 18 Jun 2008 08:01:49 GMT
View Forum Message <> Reply to Message

This allows one subsystem to require that it only be mounted when some
other subsystems are also present in the proposed hierarchy.

For example if subsystem foo depends on bar, the following will fail:
 # mount -t cgroup -ofoo xxx /dev/cgroup

You should mount with both subsystems:
 # mount -t cgroup -ofoo,bar xxx /dev/cgroup

Signed-off-by: Li Zefan <lizf@cn.fujitsu.com>

 Documentation/cgroups.txt | 6 ++++++
 include/linux/cgroup.h | 2 ++
 kernel/cgroup.c | 30 ++++++++++++++++++++++++++++++
 3 files changed, 38 insertions(+), 0 deletions(-)

diff --git a/Documentation/cgroups.txt b/Documentation/cgroups.txt
index 824fc02..8252f5b 100644
--- a/Documentation/cgroups.txt
+++ b/Documentation/cgroups.txt
@@ -530,6 +530,12 @@ and root cgroup. Currently this will only involve movement between
 the default hierarchy (which never has sub-cgroups) and a hierarchy
 that is being created/destroyed (and hence has no sub-cgroups).

+int subsys_depend(struct cgroup_subsys *ss, unsigned long subsys_bits)
+
+Called when a cgroup subsystem wants to check if some other subsystems
+are also present in the proposed hierarchy. If this method returns error,
+the mount of the cgroup filesystem will fail.
+
 4. Questions
 ============

diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h
index e155aa7..fc99ba4 100644
--- a/include/linux/cgroup.h
+++ b/include/linux/cgroup.h
@@ -305,6 +305,8 @@ struct cgroup_subsys {
     struct cgroup *cgrp);
  void (*post_clone)(struct cgroup_subsys *ss, struct cgroup *cgrp);
  void (*bind)(struct cgroup_subsys *ss, struct cgroup *root);
+ int (*subsys_depend)(struct cgroup_subsys *ss,
+   unsigned long subsys_bits);
  /*

```
   * This routine is called with the task_lock of mm->owner held
   */
diff --git a/kernel/cgroup.c b/kernel/cgroup.c
index 15ac0e1..a4c8671 100644
--- a/kernel/cgroup.c
+++ b/kernel/cgroup.c
@@ -837,6 +837,25 @@ static int parse_cgroupfs_options(char *data,
  return 0;
 }

+static int check_subsys_dependency(unsigned long subsys_bits)
+{
+ int i;
+ int ret;
+ struct cgroup_subsys *ss;
+
+ for (i = 0; i < CGROUP_SUBSYS_COUNT; i++) {
+  ss = subsys[i];
+
+  if (test_bit(i, &subsys_bits) && ss->subsys_depend) {
+   ret = ss->subsys_depend(ss, subsys_bits);
+   if (ret)
+    return ret;
+  }
+ }
+
+ return 0;
+}
+
 static int cgroup_remount(struct super_block *sb, int *flags, char *data)
 {
  int ret = 0;
@@ -852,6 +871,10 @@ static int cgroup_remount(struct super_block *sb, int *flags, char *data)
  if (ret)
   goto out_unlock;

+ ret = check_subsys_dependency(opts.subsys_bits);
+ if (ret)
+  goto out_unlock;
+
  /* Don't allow flags to change at remount */
  if (opts.flags != root->flags) {
   ret = -EINVAL;
@@ -972,6 +995,13 @@ static int cgroup_get_sb(struct file_system_type *fs_type,
   return ret;
  }

+ ret = check_subsys_dependency(opts.subsys_bits);
```

```
+ if (ret) {
+  if (opts.release_agent)
+   kfree(opts.release_agent);
+  return ret;
+ }
+
  root = kzalloc(sizeof(*root), GFP_KERNEL);
  if (!root) {
   if (opts.release_agent)
--
1.5.4.rc3
```