Subject: Re: [RFC] Control Groups Roadmap ideas
Posted by Li Zefan on Wed, 09 Apr 2008 02:28:15 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> 3) Subsystem dependencies
> -----
>
> This would be a fairly simple change, essentially allowing one
> subsystem to require that it only be mounted on a hierarchy when some
> other subsystem was also present. The implementation would probably be
> a callback that allows a subsystem to confirm whether it's prepared to
> be included in a proposed hierarchy containing a specified subsystem
> bitmask; it would be able to prevent the hierarchy from being created
> by giving an error return. An example of a use for this would be a
> swap subsystem that is mostly independent of the memory controller,
> but uses the page-ownership tracking of the memory controller to
> determine which cgroup to charge swap pages to. Hence it would require
> that it only be mounted on a hierarchy that also included a memory
> controller. The memory controller would make no such requirement by
> itself, so could be used on its own without the swap controller.
>

Sounds good, and I wrote a prototype in a quick:

```
diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h
index a6a6035..091bc21 100644
--- a/include/linux/cgroup.h
+++ b/include/linux/cgroup.h
@@ -254,6 +254,7 @@ 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 (*can_mount)(struct cgroup_subsys *ss, unsigned long subsys_bits);
  int subsys_id;
  int active;
  int disabled;
diff --git a/kernel/cgroup.c b/kernel/cgroup.c
index 62f1a52..3d43ff2 100644
--- a/kernel/cgroup.c
+++ b/kernel/cgroup.c
@@ -824,6 +824,25 @@ static int parse_cgroupfs_options(char *data,
  return 0;
 }

+static int check_mount(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->can_mount) {
+   ret = ss->can_mount(ss, subsys_bits);
+   if (ret)
+    return ret;
+  }
+ }
+
+ return 0;
+}
+
 static int cgroup_remount(struct super_block *sb, int *flags, char *data)
 {
  int ret = 0;
@@ -839,6 +858,10 @@ static int cgroup_remount(struct super_block *sb, int *flags, char *data)
  if (ret)
   goto out_unlock;

+ ret = check_mount(opts.subsys_bits);
+ if (ret)
+  goto out_unlock;
+
  /* Don't allow flags to change at remount */
  if (opts.flags != root->flags) {
   ret = -EINVAL;
@@ -959,6 +982,13 @@ static int cgroup_get_sb(struct file_system_type *fs_type,
   return ret;
  }

+ ret = check_mount(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)
-------
```

for the example about swap controller and memory controller:

```
static int swap_cgroup_can_mount(struct cgroup_subsys *ss,
    unsigned long subsys_bits)
{
 if (!test_bit(mem_cgroup_subsys_id, &subsys_bits))
  return -EINVAL;
        return 0;
}
```

'mem_cgroup_subsys_id' is a member of enum cgroup_subsys_id defined in cgroup.h

_____

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