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
```

Subject: Re: [PATCH] cgroup: support checking of subsystem dependencies
Posted by Paul Menage on Wed, 18 Jun 2008 21:13:58 GMT
View Forum Message <> Reply to Message

On Wed, Jun 18, 2008 at 1:01 AM, Li Zefan <lizf@cn.fujitsu.com> wrote:
> 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;
> +

Can we just call this from one place, in parse_cgroupfs_options() ?

Paul

Subject: Re: [PATCH] cgroup: support checking of subsystem dependencies
Posted by KAMEZAWA Hiroyuki on Thu, 19 Jun 2008 00:13:20 GMT
View Forum Message <> Reply to Message

On Wed, 18 Jun 2008 16:01:49 +0800
Li Zefan <lizf@cn.fujitsu.com> wrote:

> 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
>
I'm just curious. May I ask "Is there such cgroup subsystem now ?"

Thanks,
-Kame

> 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
>
>
>
```

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

Subject: Re: [PATCH] cgroup: support checking of subsystem dependencies
Posted by Li Zefan on Thu, 19 Jun 2008 00:40:24 GMT
View Forum Message <> Reply to Message

KAMEZAWA Hiroyuki wrote:
> On Wed, 18 Jun 2008 16:01:49 +0800
> Li Zefan <lizf@cn.fujitsu.com> wrote:
>
>> 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
>>
> I'm just curious. May I ask "Is there such cgroup subsystem now ?"
>

Currently no, but people who are developing or wants to develop new subsystems
may find this useful, for example Daisuke Nishimura-san's swap controller.

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

Subject: Re: [PATCH] cgroup: support checking of subsystem dependencies
Posted by Li Zefan on Thu, 19 Jun 2008 00:43:18 GMT
View Forum Message <> Reply to Message

>> 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;
>> +
>
> Can we just call this from one place, in parse_cgroupfs_options() ?
>

I think yes we can. I'll revise it.

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

---

Subject: [PATCH] cgroup: support checking of subsystem dependencies (v2)
Posted by Li Zefan on Thu, 19 Jun 2008 01:51:36 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

foo may implement the subsys_depend() callback this way:

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

Changelog:
- call check_subsys_depend() in parse_cgroupfs_options(), but not in mount
  and remount code.

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

 Documentation/cgroups.txt | 6 ++++++
 include/linux/cgroup.h | 2 ++

```
 kernel/cgroup.c          |   21 ++++++++++++++++++++-
 3 files changed, 28 insertions(+), 1 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..18e8132 100644
--- a/kernel/cgroup.c
+++ b/kernel/cgroup.c
@@ -774,6 +774,25 @@ static int cgroup_show_options(struct seq_file *seq, struct vfsmount
*vfs)
  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;
+}
+
 struct cgroup_sb_opts {
  unsigned long subsys_bits;
  unsigned long flags;
@@ -834,7 +853,7 @@ static int parse_cgroupfs_options(char *data,
  if (!opts->subsys_bits)
   return -EINVAL;

- return 0;
+ return check_subsys_dependency(opts->subsys_bits);
 }

 static int cgroup_remount(struct super_block *sb, int *flags, char *data)
--
1.5.4.rc3
```

Subject: Re: [PATCH] cgroup: support checking of subsystem dependencies
Posted by Daisuke Nishimura on Thu, 19 Jun 2008 02:30:58 GMT
View Forum Message <> Reply to Message

On Thu, 19 Jun 2008 08:40:24 +0800, Li Zefan <lizf@cn.fujitsu.com> wrote:
> KAMEZAWA Hiroyuki wrote:
> > On Wed, 18 Jun 2008 16:01:49 +0800
> > Li Zefan <lizf@cn.fujitsu.com> wrote:
> >
> >> 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
> >>
> > I'm just curious. May I ask "Is there such cgroup subsystem now ?"
> >
>
> Currently no, but people who are developing or wants to develop new subsystems
> may find this useful, for example Daisuke Nishimura-san's swap controller.
>
My current version is implemented as addon to memory controller,
so it can be used by just mounting memory controller :)

But anyway, this feature might be used in future by some subsystems.

Thanks,
Daisuke Nishimura.

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

---

Subject: Re: [PATCH] cgroup: support checking of subsystem dependencies (v2)
Posted by Paul Menage on Fri, 20 Jun 2008 04:51:53 GMT
View Forum Message <> Reply to Message

On Wed, Jun 18, 2008 at 6:51 PM, Li Zefan <lizf@cn.fujitsu.com> wrote:
>
> Signed-off-by: Li Zefan <lizf@cn.fujitsu.com>

Acked-by: Paul Menage <menage@google.com>

Thanks.

> ---
> Documentation/cgroups.txt | 6 ++++++
> include/linux/cgroup.h | 2 ++
> kernel/cgroup.c | 21 +++++++++++++++++++-
> 3 files changed, 28 insertions(+), 1 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..18e8132 100644
> --- a/kernel/cgroup.c
> +++ b/kernel/cgroup.c
> @@ -774,6 +774,25 @@ static int cgroup_show_options(struct seq_file *seq, struct vfsmount
*vfs)
>       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;
> +}
> +
> struct cgroup_sb_opts {
> 	unsigned long subsys_bits;
> 	unsigned long flags;
> @@ -834,7 +853,7 @@ static int parse_cgroupfs_options(char *data,
> 	if (!opts->subsys_bits)
> 		return -EINVAL;
>
> -	return 0;
> +	return check_subsys_dependency(opts->subsys_bits);
> }
>
> static int cgroup_remount(struct super_block *sb, int *flags, char *data)
> --
> 1.5.4.rc3
>
>

Subject: Re: [PATCH] cgroup: support checking of subsystem dependencies (v2)
Posted by akpm on Tue, 01 Jul 2008 07:51:40 GMT
View Forum Message <> Reply to Message

On Thu, 19 Jun 2008 09:51:36 +0800 Li Zefan <lizf@cn.fujitsu.com> wrote:

> 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
>
> foo may implement the subsys_depend() callback this way:
>
> static int foo_cgroup_subsys_depend(struct cgroup_subsys *ss,
> 		unsigned long subsys_bits)
> {
> if (!test_bit(bar_cgroup_subsys_id, &subsys_bits))

>   return -EINVAL;
> return 0;
> }
>
> Changelog:
> - call check_subsys_depend() in parse_cgroupfs_options(), but not in mount
>   and remount code.
>
> Signed-off-by: Li Zefan <lizf@cn.fujitsu.com>
> ---
> Documentation/cgroups.txt | 6 ++++++
> include/linux/cgroup.h | 2 ++
> kernel/cgroup.c | 21 +++++++++++++++++++++-
> 3 files changed, 28 insertions(+), 1 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.

OK, but the name subsys_depend is quite poor.

check_subsys_dependency is better. But it still has the failing that
the reader cannot determine the sense of the function's return value
from its name. Does it return true on success, or false?

A good name would be something like subsys_dependencies_ok(). Then
code such as

```
 if (subsys_dependencies_ok(...))
  go_wild();
 else
  bad_hair_day();
```

makes more sense.

> 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..18e8132 100644
> --- a/kernel/cgroup.c
> +++ b/kernel/cgroup.c
> @@ -774,6 +774,25 @@ static int cgroup_show_options(struct seq_file *seq, struct vfsmount
*vfs)
>   return 0;
> }
>
> +static int check_subsys_dependency(unsigned long subsys_bits)

Would be nice to have a little comment explaining this function's role
in the world. It should document the meaning of the return values.

Perhaps it could return bool. That depends upon a well-chosen name,
and upon the thus-far-undocumented return-value meaning.

> +{
> + 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;
> +}

> struct cgroup_sb_opts {
>   unsigned long subsys_bits;
>   unsigned long flags;
> @@ -834,7 +853,7 @@ static int parse_cgroupfs_options(char *data,
>   if (!opts->subsys_bits)
>   return -EINVAL;
>
> - return 0;
> + return check_subsys_dependency(opts->subsys_bits);
> }

The whole patch doesn't do anything. Perhaps there's another patch in
the pipeline somewhere which adds one or more ->subsys_depend
implementations, but I cannot find it. If so, I'd have expected this
patch to be titled [1/N].

Subject: Re: [PATCH] cgroup: support checking of subsystem dependencies (v2)
Posted by Li Zefan on Tue, 01 Jul 2008 08:43:30 GMT
View Forum Message <> Reply to Message

>> +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.
>
> OK, but the name subsys_depend is quite poor.
>
> check_subsys_dependency is better. But it still has the failing that
> the reader cannot determine the sense of the function's return value
> from its name. Does it return true on success, or false?
>

Other callbacks return -errno on failure and 0 on success, so I make this new
callback behave the same, and the errno will be returned by the mount command.

But it seems true/false is more reasonable, then if it returns false, mount can
return EINVAL.

I'll document the return value.

> A good name would be something like subsys_dependencies_ok(). Then
> code such as
>
> if (subsys_dependencies_ok(...))
>   go_wild();
> else
>   bad_hair_day();
>
> makes more sense.
>

Seems better.

>
>> 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..18e8132 100644
>> --- a/kernel/cgroup.c
>> +++ b/kernel/cgroup.c
>> @@ -774,6 +774,25 @@ static int cgroup_show_options(struct seq_file *seq, struct vfsmount
*vfs)
>>   return 0;
>> }
>>
>> +static int check_subsys_dependency(unsigned long subsys_bits)
>
> Would be nice to have a little comment explaining this function's role
> in the world. It should document the meaning of the return values.
>
> Perhaps it could return bool. That depends upon a well-chosen name,
> and upon the thus-far-undocumented return-value meaning.
>

will fix.

```
>> +{
>> + 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;
>> +}
>
>
>>  struct cgroup_sb_opts {
>>   unsigned long subsys_bits;
>>   unsigned long flags;
>> @@ -834,7 +853,7 @@ static int parse_cgroupfs_options(char *data,
>>   if (!opts->subsys_bits)
>>    return -EINVAL;
>>
>> - return 0;
>> + return check_subsys_dependency(opts->subsys_bits);
>>  }
>
> The whole patch doesn't do anything.  Perhaps there's another patch in
> the pipeline somewhere which adds one or more ->subsys_depend
> implementations, but I cannot find it.  If so, I'd have expected this
> patch to be titled [1/N].
>
```

Yes, the patch does nothing actually. Daisuke Nishimura-san's original swap controller needed to be mounted with memory controller, but the newer version makes it an addon to memory controller, so it currently doesn't need the feature this patch provides.

Or I can keep this patch until some new cgroup subsystem needs it ?

Subject: Re: [PATCH] cgroup: support checking of subsystem dependencies (v2)
Posted by akpm on Tue, 01 Jul 2008 08:54:38 GMT
View Forum Message <> Reply to Message

On Tue, 01 Jul 2008 16:43:30 +0800 Li Zefan <lizf@cn.fujitsu.com> wrote:

> > The whole patch doesn't do anything. Perhaps there's another patch in
> > the pipeline somewhere which adds one or more ->subsys_depend
> > implementations, but I cannot find it. If so, I'd have expected this
> > patch to be titled [1/N].
> >
>
> Yes, the patch does nothing actually. Daisuke Nishimura-san's original swap
> controller needed to be mounted with memory controller, but the newer
> version makes it an addon to memory controller, so it currently doesn't
> need the feature this patch provides.
>
> Or I can keep this patch until some new cgroup subsystem needs it ?

That would be better. Once something needs this patch, we do

[patch 1/n]: cgroup: support checking of subsystem dependencies
[patch 2/n]: <use it>

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