Subject: [PATCH 0/3] rcu - removing superfluous rcu_read_lock_held check
Posted by Jiri Olsa on Mon, 01 Nov 2010 19:15:06 GMT
View Forum Message <> Reply to Message

hi,

the rcu_dereference_check is defined as

```
#define rcu_dereference_check(p, c) \
    __rcu_dereference_check((p), rcu_read_lock_held() || (c), __rcu)
```

so the caller does not need to specify rcu_read_lock_held()
condition.

Several places in kernel are spcifying rcu_read_lock_held as
rcu_dereference_check condition parameter. I separated them
into 3 patches:

1/3 - cgroup - removing superfluous rcu_read_lock_held check
2/3 - kernel,cred,kvm,security - removing superfluous rcu_read_lock_held check
3/3 - net - removing superfluous rcu_read_lock_held check

sry if I might missed or added somebody wrongly to recipients

wbr,
jirka

Signed-off-by: Jiri Olsa <jolsa@redhat.com>

```
 include/linux/cgroup.h            |    1 -
 include/linux/cred.h              |    1 -
 include/linux/fdtable.h           |    1 -
 include/linux/kvm_host.h          |    1 -
 include/linux/rtnetlink.h         |    3 +--
 include/net/sock.h                |    3 +--
 kernel/cgroup.c                   |    6 ++----
 kernel/exit.c                     |    1 -
 kernel/pid.c                      |    1 -
 kernel/rcutorture.c               |    2 --
 net/mac80211/sta_info.c           |    4 ----
 net/netlabel/netlabel_domainhash.c |    3 +--
 net/netlabel/netlabel_unlabeled.c  |    3 +--
 security/keys/keyring.c           |    1 -
 14 files changed, 6 insertions(+), 25 deletions(-)
```

_____
Containers mailing list

Subject: [PATCH 1/3] cgroup - removing superfluous rcu_read_lock_held check
Posted by Jiri Olsa on Mon, 01 Nov 2010 19:15:07 GMT
View Forum Message <> Reply to Message

hi,
the rcu_dereference_check is defined as

```
 #define rcu_dereference_check(p, c) \
    __rcu_dereference_check((p), rcu_read_lock_held() || (c), __rcu)
```

so the caller does not need to specify rcu_read_lock_held() condition.

wbr,
jirka

Signed-off-by: Jiri Olsa <jolsa@redhat.com>

```
---
 include/linux/cgroup.h |    1 -
 kernel/cgroup.c        |    6 ++----
 2 files changed, 2 insertions(+), 5 deletions(-)

diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h
index ed4ba11..caed568 100644
--- a/include/linux/cgroup.h
+++ b/include/linux/cgroup.h
@@ -536,7 +536,6 @@ static inline struct cgroup_subsys_state *cgroup_subsys_state(
  */
 #define task_subsys_state_check(task, subsys_id, __c)   \
  rcu_dereference_check(task->cgroups->subsys[subsys_id],  \
-         rcu_read_lock_held() ||   \
          lockdep_is_held(&task->alloc_lock) || \
          cgroup_lock_is_held() || (__c))

diff --git a/kernel/cgroup.c b/kernel/cgroup.c
index 66a416b..1f329a2 100644
--- a/kernel/cgroup.c
+++ b/kernel/cgroup.c
@@ -1687,7 +1687,6 @@ int cgroup_path(const struct cgroup *cgrp, char *buf, int buflen)
 {
  char *start;
  struct dentry *dentry = rcu_dereference_check(cgrp->dentry,
-           rcu_read_lock_held() ||
            cgroup_lock_is_held());
```

```
  if (!dentry || cgrp == dummytop) {
@@ -1713,7 +1712,6 @@ int cgroup_path(const struct cgroup *cgrp, char *buf, int buflen)
    break;

   dentry = rcu_dereference_check(cgrp->dentry,
-           rcu_read_lock_held() ||
            cgroup_lock_is_held());
   if (!cgrp->parent)
    continue;
@@ -4544,7 +4542,7 @@ unsigned short css_id(struct cgroup_subsys_state *css)
   * it's unchanged until freed.
   */
  cssid = rcu_dereference_check(css->id,
-   rcu_read_lock_held() || atomic_read(&css->refcnt));
+   atomic_read(&css->refcnt));

  if (cssid)
   return cssid->id;
@@ -4557,7 +4555,7 @@ unsigned short css_depth(struct cgroup_subsys_state *css)
  struct css_id *cssid;

  cssid = rcu_dereference_check(css->id,
-   rcu_read_lock_held() || atomic_read(&css->refcnt));
+   atomic_read(&css->refcnt));

  if (cssid)
   return cssid->depth;
--
1.7.1
```

Subject: [PATCH 2/3] kernel, cred, kvm, security - removing superfluous rcu_read_lock_held check

Posted by Jiri Olsa on Mon, 01 Nov 2010 19:15:08 GMT

View Forum Message <> Reply to Message

hi,
the rcu_dereference_check is defined as

```
#define rcu_dereference_check(p, c) \
   __rcu_dereference_check((p), rcu_read_lock_held() || (c), __rcu)
```

so the caller does not need to specify rcu_read_lock_held() condition.

wbr,
jirka

Signed-off-by: Jiri Olsa <jolsa@redhat.com>
---
 include/linux/cred.h | 1 -
 include/linux/fdtable.h | 1 -
 include/linux/kvm_host.h | 1 -
 kernel/exit.c | 1 -
 kernel/pid.c | 1 -
 kernel/rcutorture.c | 2 --
 security/keys/keyring.c | 1 -
 7 files changed, 0 insertions(+), 8 deletions(-)

```
diff --git a/include/linux/cred.h b/include/linux/cred.h
index 4aaeab3..a6b9afc 100644
--- a/include/linux/cred.h
+++ b/include/linux/cred.h
@@ -283,7 +283,6 @@ static inline void put_cred(const struct cred *_cred)
  ({       \
   const struct task_struct *__t = (task);   \
   rcu_dereference_check(__t->real_cred,   \
-          rcu_read_lock_held() ||  \
           task_is_dead(__t));  \
  })

diff --git a/include/linux/fdtable.h b/include/linux/fdtable.h
index 133c0ba..df7e3cf 100644
--- a/include/linux/fdtable.h
+++ b/include/linux/fdtable.h
@@ -60,7 +60,6 @@ struct files_struct {

 #define rcu_dereference_check_fdtable(files, fdtfd) \
  (rcu_dereference_check((fdtfd), \
-          rcu_read_lock_held() || \
           lockdep_is_held(&(files)->file_lock) || \
           atomic_read(&(files)->count) == 1 || \
           rcu_my_thread_group_empty()))
diff --git a/include/linux/kvm_host.h b/include/linux/kvm_host.h
index a055742..a90a7e3 100644
--- a/include/linux/kvm_host.h
+++ b/include/linux/kvm_host.h
@@ -256,7 +256,6 @@ void kvm_put_kvm(struct kvm *kvm);
 static inline struct kvm_memslots *kvm_memslots(struct kvm *kvm)
 {
```

```
  return rcu_dereference_check(kvm->memslots,
-   srcu_read_lock_held(&kvm->srcu)
    || lockdep_is_held(&kvm->slots_lock));
 }

diff --git a/kernel/exit.c b/kernel/exit.c
index b194feb..f753342 100644
--- a/kernel/exit.c
+++ b/kernel/exit.c
@@ -85,7 +85,6 @@ static void __exit_signal(struct task_struct *tsk)
  struct tty_struct *uninitialized_var(tty);

  sighand = rcu_dereference_check(tsk->sighand,
-     rcu_read_lock_held() ||
      lockdep_tasklist_lock_is_held());
  spin_lock(&sighand->siglock);

diff --git a/kernel/pid.c b/kernel/pid.c
index 39b65b6..c02adda 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -402,7 +402,6 @@ struct task_struct *pid_task(struct pid *pid, enum pid_type type)
  if (pid) {
   struct hlist_node *first;
   first = rcu_dereference_check(hlist_first_rcu(&pid->tasks[type]),
-           rcu_read_lock_held() ||
           lockdep_tasklist_lock_is_held());
   if (first)
    result = hlist_entry(first, struct task_struct, pids[(type)].node);
diff --git a/kernel/rcutorture.c b/kernel/rcutorture.c
index 9d8e8fb..0956a73 100644
--- a/kernel/rcutorture.c
+++ b/kernel/rcutorture.c
@@ -807,7 +807,6 @@ static void rcu_torture_timer(unsigned long unused)
  idx = cur_ops->readlock();
  completed = cur_ops->completed();
  p = rcu_dereference_check(rcu_torture_current,
-      rcu_read_lock_held() ||
       rcu_read_lock_bh_held() ||
       rcu_read_lock_sched_held() ||
       srcu_read_lock_held(&srcu_ctl));
@@ -868,7 +867,6 @@ rcu_torture_reader(void *arg)
   idx = cur_ops->readlock();
   completed = cur_ops->completed();
   p = rcu_dereference_check(rcu_torture_current,
-       rcu_read_lock_held() ||
        rcu_read_lock_bh_held() ||
        rcu_read_lock_sched_held() ||
```

```
        srcu_read_lock_held(&srcu_ctl));
diff --git a/security/keys/keyring.c b/security/keys/keyring.c
index d37f713..73c23f2 100644
--- a/security/keys/keyring.c
+++ b/security/keys/keyring.c
@@ -157,7 +157,6 @@ static void keyring_destroy(struct key *keyring)
  }

  klist = rcu_dereference_check(keyring->payload.subscriptions,
-         rcu_read_lock_held() ||
          atomic_read(&keyring->usage) == 0);
  if (klist) {
   for (loop = klist->nkeys - 1; loop >= 0; loop--)
--
1.7.1
```

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

---

Subject: [PATCH 3/3] net - removing superfluous rcu_read_lock_held check
Posted by Jiri Olsa on Mon, 01 Nov 2010 19:15:09 GMT
View Forum Message <> Reply to Message

hi,
the rcu_dereference_check is defined as

```
 #define rcu_dereference_check(p, c) \
    __rcu_dereference_check((p), rcu_read_lock_held() || (c), __rcu)
```

so the caller does not need to specify rcu_read_lock_held() condition.

wbr,
jirka

```
Signed-off-by: Jiri Olsa <jolsa@redhat.com>
---
 include/linux/rtnetlink.h          |    3 +--
 include/net/sock.h                 |    3 +--
 net/mac80211/sta_info.c            |    4 ----
 net/netlabel/netlabel_domainhash.c |    3 +--
 net/netlabel/netlabel_unlabeled.c  |    3 +--
 5 files changed, 4 insertions(+), 12 deletions(-)

diff --git a/include/linux/rtnetlink.h b/include/linux/rtnetlink.h
```

```
index d42f274..dfe5ba1 100644
--- a/include/linux/rtnetlink.h
+++ b/include/linux/rtnetlink.h
@@ -758,8 +758,7 @@ extern int lockdep_rtnl_is_held(void);
  * or RTNL. Note : Please prefer rtnl_dereference() or rcu_dereference()
  */
 #define rcu_dereference_rtnl(p)     \
- rcu_dereference_check(p, rcu_read_lock_held() || \
-     lockdep_rtnl_is_held())
+ rcu_dereference_check(p, lockdep_rtnl_is_held())

 /**
  * rtnl_dereference - fetch RCU pointer when updates are prevented by RTNL
diff --git a/include/net/sock.h b/include/net/sock.h
index c7a7362..bee3e9c 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -1274,8 +1274,7 @@ extern unsigned long sock_i_ino(struct sock *sk);
 static inline struct dst_entry *
 __sk_dst_get(struct sock *sk)
 {
- return rcu_dereference_check(sk->sk_dst_cache, rcu_read_lock_held() ||
-             sock_owned_by_user(sk) ||
+ return rcu_dereference_check(sk->sk_dst_cache, sock_owned_by_user(sk) ||
              lockdep_is_held(&sk->sk_lock.slock));
 }

diff --git a/net/mac80211/sta_info.c b/net/mac80211/sta_info.c
index 6d8f897..c879217 100644
--- a/net/mac80211/sta_info.c
+++ b/net/mac80211/sta_info.c
@@ -94,7 +94,6 @@ struct sta_info *sta_info_get(struct ieee80211_sub_if_data *sdata,
  struct sta_info *sta;

  sta = rcu_dereference_check(local->sta_hash[STA_HASH(addr)],
-         rcu_read_lock_held() ||
          lockdep_is_held(&local->sta_lock) ||
          lockdep_is_held(&local->sta_mtx));
  while (sta) {
@@ -102,7 +101,6 @@ struct sta_info *sta_info_get(struct ieee80211_sub_if_data *sdata,
        memcmp(sta->sta.addr, addr, ETH_ALEN) == 0)
     break;
   sta = rcu_dereference_check(sta->hnext,
-          rcu_read_lock_held() ||
          lockdep_is_held(&local->sta_lock) ||
          lockdep_is_held(&local->sta_mtx));
  }
@@ -120,7 +118,6 @@ struct sta_info *sta_info_get_bss(struct ieee80211_sub_if_data *sdata,
```

```
  struct sta_info *sta;

  sta = rcu_dereference_check(local->sta_hash[STA_HASH(addr)],
-         rcu_read_lock_held() ||
          lockdep_is_held(&local->sta_lock) ||
          lockdep_is_held(&local->sta_mtx));
  while (sta) {
@@ -129,7 +126,6 @@ struct sta_info *sta_info_get_bss(struct ieee80211_sub_if_data *sdata,
        memcmp(sta->sta.addr, addr, ETH_ALEN) == 0)
     break;
    sta = rcu_dereference_check(sta->hnext,
-           rcu_read_lock_held() ||
           lockdep_is_held(&local->sta_lock) ||
           lockdep_is_held(&local->sta_mtx));
  }
diff --git a/net/netlabel/netlabel_domainhash.c b/net/netlabel/netlabel_domainhash.c
index d37b7f8..82795a4 100644
--- a/net/netlabel/netlabel_domainhash.c
+++ b/net/netlabel/netlabel_domainhash.c
@@ -55,8 +55,7 @@ struct netlbl_domhsh_tbl {
  * should be okay */
 static DEFINE_SPINLOCK(netlbl_domhsh_lock);
 #define netlbl_domhsh_rcu_deref(p) \
- rcu_dereference_check(p, rcu_read_lock_held() || \
-     lockdep_is_held(&netlbl_domhsh_lock))
+ rcu_dereference_check(p, lockdep_is_held(&netlbl_domhsh_lock))
 static struct netlbl_domhsh_tbl *netlbl_domhsh = NULL;
 static struct netlbl_dom_map *netlbl_domhsh_def = NULL;

diff --git a/net/netlabel/netlabel_unlabeled.c b/net/netlabel/netlabel_unlabeled.c
index e2b0a68..d2f982f 100644
--- a/net/netlabel/netlabel_unlabeled.c
+++ b/net/netlabel/netlabel_unlabeled.c
@@ -116,8 +116,7 @@ struct netlbl_unlhsh_walk_arg {
  * hash table should be okay */
 static DEFINE_SPINLOCK(netlbl_unlhsh_lock);
 #define netlbl_unlhsh_rcu_deref(p) \
- rcu_dereference_check(p, rcu_read_lock_held() || \
-     lockdep_is_held(&netlbl_unlhsh_lock))
+ rcu_dereference_check(p, lockdep_is_held(&netlbl_unlhsh_lock))
 static struct netlbl_unlhsh_tbl *netlbl_unlhsh = NULL;
 static struct netlbl_unlhsh_iface *netlbl_unlhsh_def = NULL;

--
1.7.1
```

_____
Containers mailing list

Subject: Re: [PATCH 2/3] kernel, cred, kvm, security - removing superfluous rcu_read_lock_held check
Posted by Paolo Bonzini on Mon, 01 Nov 2010 22:42:23 GMT
View Forum Message <> Reply to Message

On 11/01/2010 08:15 PM, Jiri Olsa wrote:
> diff --git a/include/linux/kvm_host.h b/include/linux/kvm_host.h
> index a055742..a90a7e3 100644
> --- a/include/linux/kvm_host.h
> +++ b/include/linux/kvm_host.h
> @@ -256,7 +256,6 @@ void kvm_put_kvm(struct kvm *kvm);
> static inline struct kvm_memslots *kvm_memslots(struct kvm *kvm)
> {
> return rcu_dereference_check(kvm->memslots,
> - srcu_read_lock_held(&kvm->srcu)
> || lockdep_is_held(&kvm->slots_lock));
> }
>

This is an srcu_read_lock_held, which you don't touch here:

> diff --git a/kernel/rcutorture.c b/kernel/rcutorture.c
> index 9d8e8fb..0956a73 100644
> --- a/kernel/rcutorture.c
> +++ b/kernel/rcutorture.c
> @@ -807,7 +807,6 @@ static void rcu_torture_timer(unsigned long unused)
> idx = cur_ops->readlock();
> completed = cur_ops->completed();
> p = rcu_dereference_check(rcu_torture_current,
> - rcu_read_lock_held() ||
> rcu_read_lock_bh_held() ||
> rcu_read_lock_sched_held() ||
> srcu_read_lock_held(&srcu_ctl));

I guess the kvm hunk is the incorrect one?

Paolo

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

Subject: Re: [PATCH 2/3] kernel, cred, kvm, security - removing superfluous rcu_read_lock_held check
Posted by Jiri Olsa on Tue, 02 Nov 2010 07:21:45 GMT
View Forum Message <> Reply to Message

On Mon, Nov 01, 2010 at 11:42:23PM +0100, Paolo Bonzini wrote:
> On 11/01/2010 08:15 PM, Jiri Olsa wrote:
> >diff --git a/include/linux/kvm_host.h b/include/linux/kvm_host.h
> >index a055742..a90a7e3 100644
> >--- a/include/linux/kvm_host.h
> >+++ b/include/linux/kvm_host.h
> >@@ -256,7 +256,6 @@ void kvm_put_kvm(struct kvm *kvm);
> > static inline struct kvm_memslots *kvm_memslots(struct kvm *kvm)
> > {
> > return rcu_dereference_check(kvm->memslots,
> >- srcu_read_lock_held(&kvm->srcu)
> > || lockdep_is_held(&kvm->slots_lock));
> > }
> >
>
> This is an srcu_read_lock_held, which you don't touch here:
>
> >diff --git a/kernel/rcutorture.c b/kernel/rcutorture.c
> >index 9d8e8fb..0956a73 100644
> >--- a/kernel/rcutorture.c
> >+++ b/kernel/rcutorture.c
> >@@ -807,7 +807,6 @@ static void rcu_torture_timer(unsigned long unused)
> > idx = cur_ops->readlock();
> > completed = cur_ops->completed();
> > p = rcu_dereference_check(rcu_torture_current,
> >- rcu_read_lock_held() ||
> > rcu_read_lock_bh_held() ||
> > rcu_read_lock_sched_held() ||
> > srcu_read_lock_held(&srcu_ctl));
>
> I guess the kvm hunk is the incorrect one?
ops, you're right. I overlooked it, please skip the kvm hunk..

thanks,
jirka

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

Subject: Re: [PATCH 1/3] cgroup - removing superfluous rcu_read_lock_held check
Posted by Li Zefan on Tue, 02 Nov 2010 17:54:55 GMT

> hi,

This..

> the rcu_dereference_check is defined as
>
> #define rcu_dereference_check(p, c) \
>   __rcu_dereference_check((p), rcu_read_lock_held() || (c), __rcu)
>
> so the caller does not need to specify rcu_read_lock_held() condition.
>

> wbr,
> jirka

and this should be excluded from the changelog.

>
>
> Signed-off-by: Jiri Olsa <jolsa@redhat.com>

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

However a nitpick:

> ---
> include/linux/cgroup.h | 1 -
> kernel/cgroup.c | 6 ++----
> 2 files changed, 2 insertions(+), 5 deletions(-)
...
> @@ -4544,7 +4542,7 @@ unsigned short css_id(struct cgroup_subsys_state *css)
>   * it's unchanged until freed.
>   */
>   cssid = rcu_dereference_check(css->id,
> -   rcu_read_lock_held() || atomic_read(&css->refcnt));
> +   atomic_read(&css->refcnt));

Now the 2 lines can be made into one line and still fit into 80 chars.

>
>   if (cssid)
>   return cssid->id;
> @@ -4557,7 +4555,7 @@ unsigned short css_depth(struct cgroup_subsys_state *css)
>   struct css_id *cssid;
>
>   cssid = rcu_dereference_check(css->id,

> - rcu_read_lock_held() || atomic_read(&css->refcnt));
> + atomic_read(&css->refcnt));

dito

>
> if (cssid)
> return cssid->depth;

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

---

Subject: Re: [PATCH 1/3] cgroup - removing superfluous rcu_read_lock_held check
Posted by Jiri Olsa on Wed, 03 Nov 2010 08:58:04 GMT
View Forum Message <> Reply to Message

On Wed, Nov 03, 2010 at 01:54:55AM +0800, Li Zefan wrote:

> > hi,
>
> This..
>
> > the rcu_dereference_check is defined as
> >
> > #define rcu_dereference_check(p, c) \
> > __rcu_dereference_check((p), rcu_read_lock_held() || (c), __rcu)
> >
> > so the caller does not need to specify rcu_read_lock_held() condition.
> >
>
> > wbr,
> > jirka
>
> and this should be excluded from the changelog.
>
> > >
> > >
> > > Signed-off-by: Jiri Olsa <jolsa@redhat.com>
>
> Reviewed-by: Li Zefan <lizf@cn.fujitsu.com>
>
> However a nitpick:
>
> > > ---
> > > include/linux/cgroup.h | 1 -
> > > kernel/cgroup.c | 6 ++----

> > 2 files changed, 2 insertions(+), 5 deletions(-)
> ...
> > @@ -4544,7 +4542,7 @@ unsigned short css_id(struct cgroup_subsys_state *css)
> >   * it's unchanged until freed.
> >   */
> >   cssid = rcu_dereference_check(css->id,
> > -   rcu_read_lock_held() || atomic_read(&css->refcnt));
> > +   atomic_read(&css->refcnt));
>
> Now the 2 lines can be made into one line and still fit into 80 chars.
>
> >
> >   if (cssid)
> >   return cssid->id;
> > @@ -4557,7 +4555,7 @@ unsigned short css_depth(struct cgroup_subsys_state *css)
> >   struct css_id *cssid;
> >
> >   cssid = rcu_dereference_check(css->id,
> > -   rcu_read_lock_held() || atomic_read(&css->refcnt));
> > +   atomic_read(&css->refcnt));
>
> dito
>
> >
> >   if (cssid)
> >   return cssid->depth;

attaching changed patch

wbr,
jirka

No need to specify rcu_read_lock_held() condition in rcu_dereference_check.

Signed-off-by: Jiri Olsa <jolsa@redhat.com>
---
 include/linux/cgroup.h | 1 -
 kernel/cgroup.c | 8 ++------
 2 files changed, 2 insertions(+), 7 deletions(-)

diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h
index ed4ba11..caed568 100644
--- a/include/linux/cgroup.h
+++ b/include/linux/cgroup.h
@@ -536,7 +536,6 @@ static inline struct cgroup_subsys_state *cgroup_subsys_state(
  */

```
 #define task_subsys_state_check(task, subsys_id, __c)   \
  rcu_dereference_check(task->cgroups->subsys[subsys_id],  \
-         rcu_read_lock_held() ||   \
          lockdep_is_held(&task->alloc_lock) || \
          cgroup_lock_is_held() || (__c))

diff --git a/kernel/cgroup.c b/kernel/cgroup.c
index 66a416b..a8d2221 100644
--- a/kernel/cgroup.c
+++ b/kernel/cgroup.c
@@ -1687,7 +1687,6 @@ int cgroup_path(const struct cgroup *cgrp, char *buf, int buflen)
 {
  char *start;
  struct dentry *dentry = rcu_dereference_check(cgrp->dentry,
-           rcu_read_lock_held() ||
            cgroup_lock_is_held());

  if (!dentry || cgrp == dummytop) {
@@ -1713,7 +1712,6 @@ int cgroup_path(const struct cgroup *cgrp, char *buf, int buflen)
    break;

   dentry = rcu_dereference_check(cgrp->dentry,
-           rcu_read_lock_held() ||
            cgroup_lock_is_held());
   if (!cgrp->parent)
    continue;
@@ -4543,8 +4541,7 @@ unsigned short css_id(struct cgroup_subsys_state *css)
   * on this or this is under rcu_read_lock(). Once css->id is allocated,
   * it's unchanged until freed.
   */
- cssid = rcu_dereference_check(css->id,
-   rcu_read_lock_held() || atomic_read(&css->refcnt));
+ cssid = rcu_dereference_check(css->id, atomic_read(&css->refcnt));

  if (cssid)
   return cssid->id;
@@ -4556,8 +4553,7 @@ unsigned short css_depth(struct cgroup_subsys_state *css)
 {
  struct css_id *cssid;

- cssid = rcu_dereference_check(css->id,
-   rcu_read_lock_held() || atomic_read(&css->refcnt));
+ cssid = rcu_dereference_check(css->id, atomic_read(&css->refcnt));

  if (cssid)
   return cssid->depth;
--
1.7.1
```

Subject: Re: [PATCH 1/3] cgroup - removing superfluous rcu_read_lock_held check
Posted by Li Zefan on Wed, 03 Nov 2010 18:40:03 GMT
View Forum Message <> Reply to Message

> attaching changed patch
>
> wbr,
> jirka
>

Some small notes:

- Cgroup patches are normally Cced to LKML, except those RFC
ones.

- I think a more commonly used title should be:
[..] cgroups: remove ...

- It's Andrew Morton that picks up cgroup patches, so he may
overlook this revised patch. So better resend this patch with
Andrew listed as receiver (and with Paul Menage, LKML ... Cced),
with a title like:
[PATCH v2] cgroups: remove ...

> ---
> No need to specify rcu_read_lock_held() condition in rcu_dereference_check.
>
>
> Signed-off-by: Jiri Olsa <jolsa@redhat.com>

Normally you should retain reviewed-by and other tags that
you've got previously.

So here again:

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