Subject: Re: [PATCH v5 2/2] decrement static keys on real destroy time
Posted by akpm on Thu, 17 May 2012 05:37:15 GMT
View Forum Message <> Reply to Message

On Thu, 17 May 2012 07:06:52 +0400 Glauber Costa <glommer@parallels.com> wrote:

```
> ...
> >> + else if (val != RESOURCE_MAX) {
> >> +  /*
> >> +   * ->activated needs to be written after the static_key update.
> >> +   *  This is what guarantees that the socket activation function
> >> +   *  is the last one to run. See sock_update_memcg() for details,
> >> +   *  and note that we don't mark any socket as belonging to this
> >> +   *  memcg until that flag is up.
> >> +   *
> >> +   *  We need to do this, because static_keys will span multiple
> >> +   *  sites, but we can't control their order. If we mark a socket
> >> +   *  as accounted, but the accounting functions are not patched in
> >> +   *  yet, we'll lose accounting.
> >> +   *
> >> +   *  We never race with the readers in sock_update_memcg(), because
> >> +   *  when this value change, the code to process it is not patched in
> >> +   *  yet.
> >> +   */
> >> +  if (!cg_proto->activated) {
> >> +   static_key_slow_inc(&memcg_socket_limit_enabled);
> >> +   cg_proto->activated = true;
> >> +  }
> >
> > If two threads run this code concurrently, they can both see
> > cg_proto->activated==false and they will both run
> > static_key_slow_inc().
> >
> > Hopefully there's some locking somewhere which prevents this, but it is
> > unobvious.  We should comment this, probably at the cg_proto.activated
> > definition site.  Or we should fix the bug ;)
> >
> If that happens, locking in static_key_slow_inc will prevent any damage.
> My previous version had explicit code to prevent that, but we were
> pointed out that this is already part of the static_key expectations, so
> that was dropped.
```

This makes no sense. If two threads run that code concurrently,
key->enabled gets incremented twice. Nobody anywhere has a record that
this happened so it cannot be undone. key->enabled is now in an
unknown state.