Subject: Re: [PATCH RESEND 2/2] Fix some kallsyms_lookup() vs rmmod races
Posted by Paulo Marques on Fri, 16 Mar 2007 17:16:39 GMT
View Forum Message <> Reply to Message

Ingo Molnar wrote:
> * Paulo Marques <pmarques@grupopie.com> wrote:
>
>>> looking at the problem from another angle: wouldnt this be something
>>> that would benefit from freeze_processes()/unfreeze_processes(), and
>>> hence no locking would be required?
>> I also considered this, but it seemed a little too "blunt" to stop
>> everything (including completely unrelated processes and kernel
>> threads) just to remove a module.
>
> 'just to remove a module' is very, very rare, on the timescale of most
> kernel ops. Almost no distro does it. Furthermore, because we want to do
> CPU-hotplug that way, we really want to make
> freeze_processes()/unfreeze_processes() 'instantaneous' to the human -
> and it is that already. (if it isnt in some case we can make it so)

Ok. I started to look at this approach and realized that module.c
already does this:

> ....
> static int __unlink_module(void *_mod)
> {
> struct module *mod = _mod;
> list_del(&mod->list);
> return 0;
> }
>
> /* Free a module, remove from lists, etc (must hold module mutex). */
> static void free_module(struct module *mod)
> {
> /* Delete from various lists */
> stop_machine_run(__unlink_module, mod, NR_CPUS);
> ....

However stop_machine_run doesn't seem like the right thing to do,
because users of the "modules" list don't seem to do anything to prevent
preemption. Am I missing something?

Does freeze_processes() / unfreeze_processes() solve this by only
freezing processes that have voluntarily scheduled (opposed to just
being preempted)?

--
Paulo Marques - www.grupopie.com

"The Computer made me do it."