Subject: Re: [PATCH] md: Remove broken SIGKILL support
Posted by ebiederm on Tue, 01 May 2007 06:13:45 GMT
View Forum Message <> Reply to Message

Neil Brown <neilb@suse.de> writes:

```
> On Thursday April 19, ebiederm@xmission.com wrote:
>> From: Eric W. Biederman <ebiederm@xmission.com>
>>
>> Currently md_thread calls allow_signal so it can receive a
>> SIGKILL but then does nothing with it except flush the
>> sigkill so that it not can use an interruptible sleep.
>>
>> This whole dance is silly so remove the unnecessary
>> and broken signal handling logic.
>
> (sorry of the delay in replying)
>
> You missed some related code which should help you see that it is -
> maybe - not completely 'silly' (though I confess it might be slightly
> broken).
> In md_check_recovery:
>
>  if (signal_pending(current)) {
>   if (mddev->pers->sync_request) {
>    printk(KERN_INFO "md: %s in immediate safe mode\n",
>           mdname(mddev));
>    mddev->safemode = 2;
>   }
>   flush_signals(current);
>  }
```

Thanks.

```
> The idea is that alt-sysrq-K will send SIGKILL to all processes
> including the md support threads, which will cause them to enter
> "immediate safe mode" so that the metadata will be marked clean
> immediately at every opportunity.  That way you can use alt-sysrq:
>   sync,unmount,kill,reboot
> and be fairly sure that you md array will be shut down cleanly.
>
> I'd be just as happy to link this into Unmount (aka
> do_emergency_remount), but that doesn't seem at all straight forward,
> and in any case should be done before the current code is ripped out.
>
> While we do have a reboot_notifier which tries to stop all arrays,
> I've never been comfortable with that.  A reboot really should just
> reboot...
```

>
> What I would REALLY like is for the block device to know whether it is
> open read-only or read-write.  Then I could mark it clean when it
> becomes read-only as would happen when do_emergency_remount remounts
> it read-only.
>
> I might see how hard that would be...

My goal to get signals to kernel threads out of the user space interface
especially for non-privileged processes, so everything that we do with
kernel threads can just be an unimportant implementation detail to user
space.

Eric