Subject: Re: [PATCHSET] Sysfs cleanups from Eric W. Biederman
Posted by ebiederm on Wed, 22 Aug 2007 15:51:35 GMT
View Forum Message <> Reply to Message

Tejun Heo <htejun@gmail.com> writes:

> Hello,
>
> Eric W. Biederman wrote:
>>> * fix-i_mutex-locking-in-sysfs_get_dentry patch is added at the top
>>>  and #14-Don_t-use-lookup_one_len_kern and
>>>  #15-vfs-Remove-lookup_one_len_kern are dropped.  This is because #14
>>>  contained had a bug where it might created dentry/inode for an
>>>  already deleted sysfs_dirent.  I think it's benefitial to keep
>>>  single lookup path.
>>
>> I think I disagree with the bug spotting.
>>
>> At least in net we the sysfs_rename_mutex which keeps parent
>> directories from disappearing.  Further we have a reference
>> to the leaf sysfs_dirent and are actively manipulating it, so
>> the sysfs_dirent should not disappear on us.
>
> sysfs_rename_mutex() keeps out renaming and moving not removing.  Also,
> reference prevents sysfs_dirent from being released not from being
> removed.  The different in lookup path is that it searches the children
> list - sd's are unlinked from children list on removal.

True.  I guess what I was seeing was not a code but a usages
restriction which may not be enforced.

If we are using sysfs_get_dentry the sd is not deleted because
it is alive and being used, likewise the path to it should also be
alive.  Once you add in classic unix delete behaviour allowing for
alive but deleted files this no longer applies.  But I don't think
that makes sense for sysfs, but I may be confused.

>>> * Rewrote simplify-sysfs_get_dentry patch and
>>>  #08-Implement-__sysfs_get_dentry,
>>>  #09-Move-sysfs_get_dentry-below-__sysfs_get_dentry and
>>>  #10-Rewrite-sysfs_get_dentry-in-terms-of-__sysfs_get_dentry are
>>>  omitted as __sysfs_get_dentry() isn't used by anyone.
>>
>> Right.  __sysfs_get_dentry is an optimization that has makes
>> the best case for sysfs_get_dentry O(1) instead of O(depth).
>> However this doesn't matter because sysfs_get_dentry is not
>> on any fast path and the maximum depth of sysfs directories
>> is fairly shallow and programmer controlled.

>
> The reason why sysfs_get_dentry() climbed up first then climbed down was
> not because of performance.  It was to support the original shadow
> implementation.  Because there was no reliable way to reach a leaf node
> from the root, dentries of all shadows are pinned such that they can
> serve as the starting point for dentry lookup and the climing up was to
> reach that starting point.  Now that the dentry-multiplexing shadow
> support is gone, there's no need to do the climing up.

Reasonable.  The reason I kept the climbing up was the performance
benefit I saw.

>> The only user other user of __sysfs_get_dentry is in the tagged
>> directory support, and even that user doesn't strictly need it.
>> Although it is a bit silly to populate the dcache just so you
>> can invalidate it a moment later...
>
> Yeah, actually the only essential user of that kind of look up is
> sysfs_drop_dentry() which is in deletion path and can't fail due to
> allocation failure and has the logic open-coded.  I have nothing against
> __sysfs_get_dentry().  It was just not needed by the patches I forwarded
> this time.  Feel free to include it as you see fit.

Sure.

> I'm currently working on kobj/sysfs separation.  As most internal
> implementation is sd based already, the changes are mostly confined to
> interface functions but it's still a big change.  I think I'll be able
> to post the patches in this week or early next week at the latest.

Ok.

I'd like to make certain we have a path to tagged support or something
like it if we can.

Eric

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