Subject: Re: [PATCH 06/11] sysfs: Implement sysfs tagged directory support.
Posted by ebiederm on Mon, 30 Jun 2008 18:56:44 GMT
View Forum Message <> Reply to Message

Tejun Heo <htejun@gmail.com> writes:

> Hello, Eric.
>
> Eric W. Biederman wrote:
>> Tejun thank you for the review, and my apologies for the delayed
>> reply.
>
> Me being the king of delays, no need for apologies. :-)
>
>>> As before, I can't bring myself to like this interface. Is computing
>>> tags dynamically really necessary? Can't we do the followings?
>>
>> It isn't so much computing tags dynamically but rather it is reading them
>> from where they are stored.
>
> It's still dynamic from sysfs's POV and I think that will make
> maintenance more difficult.

Potentially. I have no problem make it clear that things are more static.

>> There is also a second dimension here we multiplex different
>> directories based on different sets of tags. One directory based
>> on user namespaces another on the network namespaces.
>
> No matter which criteria is used to select ns, it should end up being
> mapped to a set of tags (here, ida allocated numbers). Unless tags can
> change dynamically, there shouldn't be functional difference.
>
>> The tags in practice are just pointers to the namespace pointers.
>>
>> So while we could use the ida technique to specify which set of tags
>> we are talking about for a directory it isn't sufficient.
>
> I failed to follow here. Can you please elaborate a bit? If you can
> describe a simple example to me, it would be much appreciated.

See below.

>> The question sysfs_tag_enabled(sb, tag) makes no sense to me.
>> Especially in the context of needed a sysfs_sb_show_tag(sb, tag);
>>
>> The current structure is because of all of the darn fool races and
>> magic that sysfs does. We have to say for a given directory: Your

>> contents will always be tagged, and only those that one tag that
>> matches what was captured by the superblock when sysfs is mounted
>> will be shown.
>
> sysfs_tag_enabled() was meant to test whether a directory which is
> tagged should be shown under the current sb.

Ah. When we are doing readdir or lookup. Yes that makes sense.

See below. I honestly think sysfs_tab_enabled is the wrong question.

>>> Tags which can change dynamically seems too confusing to me and it
>>> makes things difficult to verify as it's unclear how those tags are
>>> gonna to change.
>>
>> We have a fundamental issue that we have to handle, and it sounds like
>> you are proposing something that will not handle it.
>>
>> - network devices can move between namespaces.
>> - network devices have driver specific sysfs attributes hanging off of them.
>>
>> So we have to move the network devices and their sysfs attributes
>> between namespaces, and I implemented that in kobject_rename,
>> sysfs_rename path.
>>
>> The tags on a kobject can only change during a rename operation.
>> So when the change happens is well defined. Further there is a
>> set of functions: sysfs_creation_tag, sysfs_removal_tag,
>> sysfs_lookup_tag, sysfs_dirent_tag which makes it clear what we
>> are doing.
>>
>> If you really don't like how the tags are managed we need to talk
>> about how we store the tags on kobjects and on the super block.
>>
>> Registering a set of tags could easily make the sb_tag function
>> obsolete, and that is one small piece of code so it is no big deal.
>>
>> struct sysfs_tag_type_operations {
>> const void *(*mount_tag)(void);
>>   const void *(*kobject_tag)(struct kobject *kobj);
>> };
>>
>> Then we could do:
>> struct sysfs_sbtag_operations *tag_type_ops[MAX_TAG_TYPES];
>>
>> And sysfs_tag_info could become.
>> struct sysfs_tag_info {
>>   void *tag[MAX_TAG_TYPES];

>> };
>>
>> During subsystem initialization we could call
>> tag_type = sysfs_allocate_tag_type();
>>
>> Just after the subsystem creates a directory.
>> sysfs_enable_tagging(kobj/sd, tag_type);
>>
>> Then anytime we currently call sb_tag during lookup we can instead
>> just look at sysfs_info(sb)->tag[tag_type] and compare that with
>> sd->s_tag.tag.
>
> What you described is pretty much what I'm talking about. The only
> difference is whether to use caller-provided pointer as tag or an
> ida-allocated integer. The last sentence of the above paragraph is
> basically sys_tag_enabled() function (maybe misnamed).

So some concrete code examples here. In the current code in lookup
what I am doing is:

```
 tag = sysfs_lookup_tag(parent_sd, parent->d_sb);
 sd = sysfs_find_dirent(parent_sd, tag, dentry->d_name.name);
```

With the proposed change of adding tag types sysfs_lookup_tag becomes:

```
const void *sysfs_lookup_tag(struct sysfs_dirent *dir_sd, struct super_block *sb)
{
 const void *tag = NULL;

 if (dir_sd->s_flags & SYSFS_FLAG_TAGGED)
  tag = sysfs_info(sb)->tag[dir_sd->tag_type];

 return tag;
}
```

Which means that in practice I can lookup that tag that I am displaying
once.

Then in sysfs_find_dirent we do:

```
 for (sd = parent_sd->s_dir.children; sd; sd = sd->s_sibling) {
   if ((parent_sd->s_flags & SYSFS_FLAG_TAGGED) &&
       (sd->s_tag.tag != tag))
    continue;
   if (!strcmp(sd->s_name, name))
    return sd;
 }
```

That should keep the implementation sufficiently inside of sysfs for there to be no guessing. In addition as a practical matter we can only allow one tag to be visible in a directory at once or else we can not check for duplicate names. Which is the problem I see with a bitmap based test too unnecessary many degrees of freedom.

The number of tag types will be low as it is the number of subsystems that use the feature. Simple enough that I expect statically allocating the tag types in an enumeration is a safe and sane way to operate. i.e.

```
enum sysfs_tag_types {
 SYSFS_TAG_NETNS,
 SYSFS_TAG_USERNS,
       SYSFS_TAG_MAX
};
```

> The main reason why I'm whining about this so much is because I think
> tag should be something abstracted inside sysfs proper. It's something
> which affects very internal operation of sysfs and I really want to keep
> the implementation details inside sysfs. Spreading implementation over
> kobject and sysfs didn't turn out too pretty after all.

I agree. Most of the implementation is in sysfs already. We just have a few corner cases.

Fundamentally it is the subsystems responsibility that creates the kobjects and the sysfs entries. The only case where I can see an ida generated number being a help is if we start having lifetime issues. Further the extra work to allocate and free tags ida based tags seems unnecessary.

I don't doubt that there is a lot we can do better. My current goal is for something that is clean enough it won't get us into trouble later, and then merging the code. In tree where people can see the code and the interactions I expect it will be easier to talk about.

Currently the interface with the users is very small. Adding the tag_type enumeration should make it smaller and make things more obviously static.

Guys can we please make something useful happen?

Eric

Containers mailing list
Containers@lists.linux-foundation.org