Subject: Re: alternative approached at tagged nodes
Posted by Tejun Heo on Wed, 08 Aug 2007 16:31:53 GMT
View Forum Message <> Reply to Message

Okay, here's a different implementation of tagged nodes. I just compile tested it so I can guarantee it's broken. I don't think there's anything fundamentally wrong but I'm wrong fairly often, so if you find something moronic, feel free to scream at me.

1. there's no enable_tagging() or fundamental restrictions on which types of nodes can be tagged.

2. no callback. tags are set by sysfs_rename_dir_tagged(). symlinks follow the tag of its target_sd. This limits taggable node types to dirs and symlinks.

3. in all paths between the root and leaf nodes, only zero or one tagged sd exists. all children of a tagged sd have NULL s_tag but are considered tagged the same as the tagged ancestor.

4. untagged entries are visible in all supers unless there's a matching tagged entry overshadowing it. tagged entry is only visible in the matching tagged super.

5. due to #3, children in a directory which belong to the same tag can be looked up using NULL tag. Most leaf node ops don't need to be changed.

6. symlink removal is different. we either need to modify the interface to take target_sd or implement new one. actually, I think linking symlinks to target_sd and renaming/removing them automagically would be pretty good.

Thanks. What do you think?

JUST FOR BRAIN STORMING. DO NOT APPLY.

```
---
 fs/sysfs/bin.c       |    2
 fs/sysfs/dir.c       |  122 +++++++++++++++++++++++++++++++++++++++++++----------------
 fs/sysfs/file.c      |    4 -
 fs/sysfs/group.c     |    2
 fs/sysfs/inode.c     |    5 +-
 fs/sysfs/mount.c     |   44 ++++++++++++++++--
 fs/sysfs/symlink.c   |    8 ++-
 fs/sysfs/sysfs.h     |   11 +++-
 include/linux/sysfs.h |    4 +
 9 files changed, 167 insertions(+), 35 deletions(-)
```

```
Index: work/fs/sysfs/dir.c
===================================================================
--- work.orig/fs/sysfs/dir.c
+++ work/fs/sysfs/dir.c
@@ -387,7 +387,7 @@ void sysfs_addrm_start(struct sysfs_addr
  */
 int sysfs_add_one(struct sysfs_addrm_cxt *acxt, struct sysfs_dirent *sd)
 {
- if (sysfs_find_dirent(acxt->parent_sd, sd->s_name))
+ if (__sysfs_find_dirent(acxt->parent_sd, sd->s_name, sd->s_tag, 0))
   return -EEXIST;

  sd->s_parent = sysfs_get(acxt->parent_sd);
@@ -526,6 +526,24 @@ void sysfs_addrm_finish(struct sysfs_add
  }
 }

+struct sysfs_dirent *__sysfs_find_dirent(struct sysfs_dirent *parent_sd,
+      const unsigned char *name,
+      const void *tag, int match_null)
+{
+ struct sysfs_dirent *null_sd = NULL, *sd;
+
+ for (sd = parent_sd->s_children; sd; sd = sd->s_sibling) {
+  if (strcmp(sd->s_name, name))
+   continue;
+  if (sd->s_tag == tag)
+   return sd;
+  if (!sd->s_tag && match_null)
+   null_sd = sd;
+ }
+
+ return null_sd;
+}
+
 /**
  * sysfs_find_dirent - find sysfs_dirent with the given name
  * @parent_sd: sysfs_dirent to search under
@@ -542,12 +560,7 @@ void sysfs_addrm_finish(struct sysfs_add
 struct sysfs_dirent *sysfs_find_dirent(struct sysfs_dirent *parent_sd,
           const unsigned char *name)
 {
- struct sysfs_dirent *sd;
-
- for (sd = parent_sd->s_children; sd; sd = sd->s_sibling)
-  if (!strcmp(sd->s_name, name))
-   return sd;
- return NULL;
```

```
+ return __sysfs_find_dirent(parent_sd, name, NULL, 0);
 }

 /**
@@ -642,7 +655,8 @@ static struct dentry * sysfs_lookup(stru

  mutex_lock(&sysfs_mutex);

- sd = sysfs_find_dirent(parent_sd, dentry->d_name.name);
+ sd = __sysfs_find_dirent(parent_sd, dentry->d_name.name,
+      sysfs_info(dentry->d_sb)->tag, 1);

  /* no such entry */
  if (!sd)
@@ -803,9 +817,19 @@ out:
  return dentry;
 }

+const void *sysfs_dirent_tag(struct sysfs_dirent *sd)
+{
+ while (sd) {
+  if (sd->s_tag)
+   return sd->s_tag;
+  sd = sd->s_parent;
+ }
+ return sd->s_tag;
+}
+
 /**
  * sysfs_get_dentry - get dentry for the given sysfs_dirent
- * @sb: superblock of the dentry to return
+ * @super: superblock of the dentry to return
  * @sd: sysfs_dirent of interest
  *
  * Get dentry for @sd.  Dentry is looked up if currently not
@@ -820,16 +844,23 @@ out:
  * Pointer to found dentry on success, ERR_PTR() value on error.
  * NULL if the sysfs dentry does not appear in the specified superblock
  */
-struct dentry *sysfs_get_dentry(struct super_block *sb, struct sysfs_dirent *sd)
+struct dentry *sysfs_get_dentry(struct super_block *super,
+     struct sysfs_dirent *sd)
 {
+ const void *tag;
  struct sysfs_dirent *cur;
  struct dentry *parent_dentry, *dentry;

+ /* bail if this sd won't show up in this superblock */
```

```
+ tag = sysfs_dirent_tag(sd);
+ if (tag && tag != sysfs_info(super)->tag)
+  return ERR_PTR(-ENOENT);
+
  /* Find the first parent which has valid dentry.
   */
  dentry = NULL;
  cur = sd;
- while (!(dentry = __sysfs_get_dentry(sb, cur))) {
+ while (!(dentry = __sysfs_get_dentry(super, cur))) {
   if (cur->s_flags & SYSFS_FLAG_REMOVED) {
    dentry = ERR_PTR(-ENOENT);
    break;
@@ -926,27 +957,44 @@ err_out:
  return error;
 }

-
-int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
+int sysfs_rename_dir_tagged(struct kobject * kobj, const char *new_name,
+       const void *new_tag)
 {
  struct sysfs_dirent *sd = kobj->sd;
  struct list_head todo;
  struct sysfs_rename_struct *srs;
  struct inode *parent_inode = NULL;
  const char *dup_name = NULL;
- int error;
+ const void *old_tag;
+ int tag = 0, error;

  INIT_LIST_HEAD(&todo);
  mutex_lock(&sysfs_rename_mutex);

+ old_tag = sysfs_dirent_tag(sd);
+
+ /* need tagging? */
+ if (old_tag != new_tag) {
+  tag = 1;
+
+  /* can't tag again in tagged subtree */
+  error = -EINVAL;
+  if (old_tag != new_tag)
+   goto out;
+ } else
+  /* sd->s_tag can be either old_tag or NULL */
+  new_tag = sd->s_tag;
+
```

```
  error = 0;
- if (strcmp(sd->s_name, new_name) == 0)
+ if (!tag && (strcmp(sd->s_name, new_name) == 0))
    goto out; /* nothing to rename */

  sysfs_grab_supers();
- error = prep_rename(&todo, sd, sd->s_parent, new_name);
- if (error)
-  goto out_release;
+ if (!tag) {
+  error = prep_rename(&todo, sd, sd->s_parent, new_name);
+  if (error)
+   goto out_release;
+ }

  error = -ENOMEM;
  mutex_lock(&sysfs_mutex);
@@ -959,7 +1007,7 @@ int sysfs_rename_dir(struct kobject * ko
  mutex_lock(&sysfs_mutex);

  error = -EEXIST;
- if (sysfs_find_dirent(sd->s_parent, new_name))
+ if (__sysfs_find_dirent(sd->s_parent, new_name, new_tag, 0))
    goto out_unlock;

  /* rename kobject and sysfs_dirent */
@@ -974,6 +1022,7 @@ int sysfs_rename_dir(struct kobject * ko

  dup_name = sd->s_name;
  sd->s_name = new_name;
+ sd->s_tag = new_tag;

  /* rename */
  list_for_each_entry(srs, &todo, list) {
@@ -981,6 +1030,21 @@ int sysfs_rename_dir(struct kobject * ko
    d_move(srs->old_dentry, srs->new_dentry);
  }

+ /* If we are moving across superblocks drop the dcache entries */
+ if (tag) {
+  struct super_block *super;
+  struct dentry *dentry;
+
+  list_for_each_entry(super, &sysfs_fs_type.fs_supers, s_instances) {
+   dentry = __sysfs_get_dentry(super, sd);
+   if (!dentry)
+    continue;
+   shrink_dcache_parent(dentry);
```

```
+   d_drop(dentry);
+   dput(dentry);
+  }
+ }
+
  error = 0;
 out_unlock:
  mutex_unlock(&sysfs_mutex);
@@ -995,6 +1059,11 @@ out:
  return error;
 }

+int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
+{
+ return sysfs_rename_dir_tagged(kobj, new_name, kobj->sd->s_tag);
+}
+
 int sysfs_move_dir(struct kobject *kobj, struct kobject *new_parent_kobj)
 {
  struct sysfs_dirent *sd = kobj->sd;
@@ -1013,6 +1082,11 @@ int sysfs_move_dir(struct kobject *kobj,
  if (sd->s_parent == new_parent_sd)
   goto out; /* nothing to move */

+ /* can't move between parents belonging to different tags */
+ error = -EINVAL;
+ if (sysfs_dirent_tag(sd->s_parent) != sysfs_dirent_tag(new_parent_sd))
+  goto out;
+
  sysfs_grab_supers();
  error = prep_rename(&todo, sd, new_parent_sd, sd->s_name);
  if (error)
@@ -1041,7 +1115,7 @@ again:
  mutex_lock(&sysfs_mutex);

  error = -EEXIST;
- if (sysfs_find_dirent(new_parent_sd, sd->s_name))
+ if (__sysfs_find_dirent(new_parent_sd, sd->s_name, sd->s_tag, 0))
   goto out_unlock;

  error = 0;
@@ -1080,8 +1154,9 @@ static inline unsigned char dt_type(stru

 static int sysfs_readdir(struct file * filp, void * dirent, filldir_t filldir)
 {
- struct dentry *dentry = filp->f_path.dentry;
- struct sysfs_dirent * parent_sd = dentry->d_fsdata;
+ struct dentry *parent = filp->f_path.dentry;
```

```
+ struct sysfs_dirent * parent_sd = parent->d_fsdata;
+ const void *tag = sysfs_info(filp->f_dentry->d_sb)->tag;
  struct sysfs_dirent *pos;
  ino_t ino;

@@ -1110,6 +1185,9 @@ static int sysfs_readdir(struct file * f
    const char * name;
    int len;

+   if (pos->s_tag && pos->s_tag != tag)
+    continue;
+
    name = pos->s_name;
    len = strlen(name);
    filp->f_pos = ino = pos->s_ino;
Index: work/fs/sysfs/sysfs.h
===================================================================
--- work.orig/fs/sysfs/sysfs.h
+++ work/fs/sysfs/sysfs.h
@@ -26,6 +26,7 @@ struct sysfs_dirent {
  struct sysfs_dirent * s_sibling;
  struct sysfs_dirent * s_children;
  const char  * s_name;
+ const void  * s_tag;

  union {
   struct sysfs_elem_dir  dir;
@@ -51,7 +52,8 @@ struct sysfs_addrm_cxt {
 };

 struct sysfs_super_info {
- int grabbed;
+ int  grabbed;
+ const void *tag;
 };

 #define sysfs_info(SB) ((struct sysfs_super_info *)(SB)->s_fs_info)
@@ -63,6 +65,7 @@ extern struct file_system_type sysfs_fs_
 void sysfs_grab_supers(void);
 void sysfs_release_supers(void);

+extern const void *sysfs_dirent_tag(struct sysfs_dirent *sd);
 extern struct dentry *sysfs_get_dentry(struct super_block *sb, struct sysfs_dirent *sd);
 extern struct sysfs_dirent *sysfs_get_active(struct sysfs_dirent *sd);
 extern void sysfs_put_active(struct sysfs_dirent *sd);
@@ -79,6 +82,9 @@ extern void sysfs_addrm_finish(struct sy
 extern struct inode * sysfs_get_inode(struct sysfs_dirent *sd);
```

```
 extern void release_sysfs_dirent(struct sysfs_dirent * sd);
+extern struct sysfs_dirent *__sysfs_find_dirent(struct sysfs_dirent *parent_sd,
+      const unsigned char *name,
+      const void *tag, int match_null);
 extern struct sysfs_dirent *sysfs_find_dirent(struct sysfs_dirent *parent_sd,
           const unsigned char *name);
 extern struct sysfs_dirent *sysfs_get_dirent(struct sysfs_dirent *parent_sd,
@@ -88,7 +94,8 @@ extern struct sysfs_dirent *sysfs_new_di

 extern int sysfs_add_file(struct sysfs_dirent *dir_sd,
      const struct attribute *attr, int type);
-extern int sysfs_hash_and_remove(struct sysfs_dirent *dir_sd, const char *name);
+extern int sysfs_hash_and_remove(struct sysfs_dirent *dir_sd, const char *name,
+     const void *tag);
 extern struct sysfs_dirent *sysfs_find(struct sysfs_dirent *dir, const char * name);

 extern int sysfs_create_subdir(struct kobject *kobj, const char *name,
Index: work/fs/sysfs/bin.c
===================================================================
--- work.orig/fs/sysfs/bin.c
+++ work/fs/sysfs/bin.c
@@ -248,7 +248,7 @@ int sysfs_create_bin_file(struct kobject

 void sysfs_remove_bin_file(struct kobject * kobj, struct bin_attribute * attr)
 {
- if (sysfs_hash_and_remove(kobj->sd, attr->attr.name) < 0) {
+ if (sysfs_hash_and_remove(kobj->sd, attr->attr.name, NULL) < 0) {
   printk(KERN_ERR "%s: "
    "bad dentry or inode or no such file: \"%s\"\n",
    __FUNCTION__, attr->attr.name);
Index: work/fs/sysfs/file.c
===================================================================
--- work.orig/fs/sysfs/file.c
+++ work/fs/sysfs/file.c
@@ -560,7 +560,7 @@ EXPORT_SYMBOL_GPL(sysfs_chmod_file);

 void sysfs_remove_file(struct kobject * kobj, const struct attribute * attr)
 {
- sysfs_hash_and_remove(kobj->sd, attr->name);
+ sysfs_hash_and_remove(kobj->sd, attr->name, NULL);
 }


@@ -577,7 +577,7 @@ void sysfs_remove_file_from_group(struct

  dir_sd = sysfs_get_dirent(kobj->sd, group);
  if (dir_sd) {
-  sysfs_hash_and_remove(dir_sd, attr->name);
```

```
+  sysfs_hash_and_remove(dir_sd, attr->name, NULL);
   sysfs_put(dir_sd);
  }
 }
Index: work/fs/sysfs/group.c
===================================================================
--- work.orig/fs/sysfs/group.c
+++ work/fs/sysfs/group.c
@@ -23,7 +23,7 @@ static void remove_files(struct sysfs_di
  struct attribute *const* attr;

  for (attr = grp->attrs; *attr; attr++)
-  sysfs_hash_and_remove(dir_sd, (*attr)->name);
+  sysfs_hash_and_remove(dir_sd, (*attr)->name, NULL);
 }

 static int create_files(struct sysfs_dirent *dir_sd,
Index: work/fs/sysfs/inode.c
===================================================================
--- work.orig/fs/sysfs/inode.c
+++ work/fs/sysfs/inode.c
@@ -212,7 +212,8 @@ struct inode * sysfs_get_inode(struct sy
  return inode;
 }

-int sysfs_hash_and_remove(struct sysfs_dirent *dir_sd, const char *name)
+int sysfs_hash_and_remove(struct sysfs_dirent *dir_sd, const char *name,
+     const void *tag)
 {
  struct sysfs_addrm_cxt acxt;
  struct sysfs_dirent *sd;
@@ -222,7 +223,7 @@ int sysfs_hash_and_remove(struct sysfs_d

  sysfs_addrm_start(&acxt, dir_sd);

- sd = sysfs_find_dirent(dir_sd, name);
+ sd = __sysfs_find_dirent(dir_sd, name, tag, 0);
  if (sd)
   sysfs_remove_one(&acxt, sd);

Index: work/fs/sysfs/mount.c
===================================================================
--- work.orig/fs/sysfs/mount.c
+++ work/fs/sysfs/mount.c
@@ -67,6 +67,7 @@ static int sysfs_fill_super(struct super
   goto out_err;
  }
  root->d_fsdata = &sysfs_root;
```

```
+ root->d_sb = sb;
  sb->s_root = root;
  sb->s_fs_info = info;
  return 0;
@@ -80,20 +81,55 @@ out_err:
  return error;
 }

+static int sysfs_test_super(struct super_block *sb, void *ptr)
+{
+ struct task_struct *task = ptr;
+ struct sysfs_super_info *info = sysfs_info(sb);
+ int found = 1;
+
+ return found;
+}
+
 static int sysfs_get_sb(struct file_system_type *fs_type,
  int flags, const char *dev_name, void *data, struct vfsmount *mnt)
 {
- int rc;
+ struct super_block *sb;
+ int error;
  mutex_lock(&sysfs_rename_mutex);
- rc = get_sb_single(fs_type, flags, data, sysfs_fill_super, mnt);
+ sb = sget(fs_type, sysfs_test_super, set_anon_super, current);
+ if (IS_ERR(sb)) {
+  error = PTR_ERR(sb);
+  goto out;
+ }
+ if (!sb->s_root) {
+  sb->s_flags = flags;
+  error = sysfs_fill_super(sb, data, flags & MS_SILENT ? 1 : 0);
+  if (error) {
+   up_write(&sb->s_umount);
+   deactivate_super(sb);
+   goto out;
+  }
+  sb->s_flags |= MS_ACTIVE;
+ }
+ do_remount_sb(sb, flags, data, 0);
+ error = simple_set_mnt(mnt, sb);
+out:
  mutex_unlock(&sysfs_rename_mutex);
- return rc;
+ return error;
+}
+
```

```
+static void sysfs_kill_sb(struct super_block *sb)
+{
+ struct sysfs_super_info *info = sysfs_info(sb);
+
+ kill_anon_super(sb);
+ kfree(info);
 }

 struct file_system_type sysfs_fs_type = {
  .name  = "sysfs",
  .get_sb  = sysfs_get_sb,
- .kill_sb = kill_anon_super,
+ .kill_sb = sysfs_kill_sb,
 };

 void sysfs_grab_supers(void)
Index: work/fs/sysfs/symlink.c
===================================================================
--- work.orig/fs/sysfs/symlink.c
+++ work/fs/sysfs/symlink.c
@@ -87,6 +87,7 @@ int sysfs_create_link(struct kobject * k
   goto out_put;

  sd->s_elem.symlink.target_sd = target_sd;
+ sd->s_tag = sysfs_dirent_tag(target_sd);
  target_sd = NULL; /* reference is now owned by the symlink */

  sysfs_addrm_start(&acxt, parent_sd);
@@ -113,9 +114,14 @@ int sysfs_create_link(struct kobject * k

 void sysfs_remove_link(struct kobject * kobj, const char * name)
 {
- sysfs_hash_and_remove(kobj->sd, name);
+ sysfs_hash_and_remove(kobj->sd, name, NULL);
 }

+/**
+ * XXX - we need
+ * sysfs_delete_link(struct kobject *kobj, struct kobject *target);
+ */
+
 static int sysfs_get_target_path(struct sysfs_dirent * parent_sd,
      struct sysfs_dirent * target_sd, char *path)
 {
Index: work/include/linux/sysfs.h
===================================================================
--- work.orig/include/linux/sysfs.h
+++ work/include/linux/sysfs.h
```

```
@@ -97,6 +97,10 @@ extern void
 sysfs_remove_dir(struct kobject *);

 extern int __must_check
+sysfs_rename_dir_tagged(struct kobject *kobj, const char *new_name,
+   const void *new_tag);
+
+extern int __must_check
 sysfs_rename_dir(struct kobject *kobj, const char *new_name);

 extern int __must_check
```

_____

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