Subject: [PATCH] Clean proto_(un)register from in-code ifdefs
Posted by Pavel Emelianov on Tue, 06 Nov 2007 17:20:24 GMT
View Forum Message <> Reply to Message

The struct proto has the per-cpu "inuse" counter, which is handled
with a special care. All the handling code hides under the ifdef
CONFIG_SMP and it introduces some code duplication and makes it
look worse than it could.

Clean this.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/net/core/sock.c b/net/core/sock.c
index e077f26..8fc2f84 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -1819,23 +1819,48 @@ static int inuse_get(const struct proto *prot)
   res += per_cpu_ptr(prot->inuse_ptr, cpu)[0];
  return res;
 }
-#endif

-int proto_register(struct proto *prot, int alloc_slab)
+static int inuse_init(struct proto *prot)
 {
- char *request_sock_slab_name = NULL;
- char *timewait_sock_slab_name;
- int rc = -ENOBUFS;
-
-#ifdef CONFIG_SMP
  if (!prot->inuse_getval || !prot->inuse_add) {
   prot->inuse_ptr = alloc_percpu(int);
   if (prot->inuse_ptr == NULL)
-   goto out;
+   return -ENOBUFS;
+
   prot->inuse_getval = inuse_get;
   prot->inuse_add = inuse_add;
  }
+ return 0;
+}
+
+static void inuse_fini(struct proto *prot)
+{
+ if (prot->inuse_ptr != NULL) {
```

```
+  free_percpu(prot->inuse_ptr);
+  prot->inuse_ptr = NULL;
+  prot->inuse_getval = NULL;
+  prot->inuse_add = NULL;
+ }
+}
+#else
+static inline int inuse_init(struct proto *prot)
+{
+ return 0;
+}
+
+static inline void inuse_fini(struct proto *prot)
+{
+}
 #endif
+
+int proto_register(struct proto *prot, int alloc_slab)
+{
+ char *request_sock_slab_name = NULL;
+ char *timewait_sock_slab_name;
+
+ if (inuse_init(prot))
+  goto out;
+
  if (alloc_slab) {
   prot->slab = kmem_cache_create(prot->name, prot->obj_size, 0,
          SLAB_HWCACHE_ALIGN, NULL);
@@ -1887,9 +1912,8 @@ int proto_register(struct proto *prot, int alloc_slab)
  write_lock(&proto_list_lock);
  list_add(&prot->node, &proto_list);
  write_unlock(&proto_list_lock);
- rc = 0;
-out:
- return rc;
+ return 0;
+
 out_free_timewait_sock_slab_name:
  kfree(timewait_sock_slab_name);
 out_free_request_sock_slab:
@@ -1903,15 +1927,9 @@ out_free_sock_slab:
  kmem_cache_destroy(prot->slab);
  prot->slab = NULL;
 out_free_inuse:
-#ifdef CONFIG_SMP
- if (prot->inuse_ptr != NULL) {
-  free_percpu(prot->inuse_ptr);
-  prot->inuse_ptr = NULL;
```

```
-  prot->inuse_getval = NULL;
-  prot->inuse_add = NULL;
- }
-#endif
- goto out;
+ inuse_fini(prot);
+out:
+ return -ENOBUFS;
 }

 EXPORT_SYMBOL(proto_register);
@@ -1922,14 +1940,7 @@ void proto_unregister(struct proto *prot)
  list_del(&prot->node);
  write_unlock(&proto_list_lock);

-#ifdef CONFIG_SMP
- if (prot->inuse_ptr != NULL) {
-  free_percpu(prot->inuse_ptr);
-  prot->inuse_ptr = NULL;
-  prot->inuse_getval = NULL;
-  prot->inuse_add = NULL;
- }
-#endif
+ inuse_fini(prot);
  if (prot->slab != NULL) {
   kmem_cache_destroy(prot->slab);
   prot->slab = NULL;
```