

---

Subject: [PATCH 3/8] Cleanup the allocation/freeing of the sock object

Posted by [Pavel Emelianov](#) on Wed, 31 Oct 2007 12:45:03 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

The sock object is allocated either from the generic cache with the kcalloc, or from the proc->slab cache.

Move this logic into an isolated set of helpers and make the sk\_alloc/sk\_free look a bit nicer.

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

---

diff --git a/net/core/sock.c b/net/core/sock.c

index 9c2dbfa..7c2e3db 100644

--- a/net/core/sock.c

+++ b/net/core/sock.c

@@ -870,6 +870,31 @@ static void sock\_copy(struct sock \*nsk, const struct sock \*osk)  
#endif  
}

+static struct sock \*sk\_prot\_alloc(struct proto \*prot, gfp\_t priority)

+{

+ struct sock \*sk;

+ struct kmem\_cache \*slab;

+

+ slab = prot->slab;

+ if (slab != NULL)

+ sk = kmem\_cache\_alloc(slab, priority);

+ else

+ sk = kcalloc(prot->obj\_size, priority);

+

+ return sk;

+}

+

+static void sk\_prot\_free(struct proto \*prot, struct sock \*sk)

+{

+ struct kmem\_cache \*slab;

+

+ slab = prot->slab;

+ if (slab != NULL)

+ kmem\_cache\_free(slab, sk);

+ else

+ kfree(sk);

+}

+

/\*\*

```

* sk_alloc - All socket objects are allocated here
* @net: the applicable net namespace
@@ -881,14 +906,9 @@ static void sock_copy(struct sock *nsk, const struct sock *osk)
struct sock *sk_alloc(struct net *net, int family, gfp_t priority,
    struct proto *prot, int zero_it)
{
- struct sock *sk = NULL;
- struct kmem_cache *slab = prot->slab;
-
- if (slab != NULL)
-   sk = kmem_cache_alloc(slab, priority);
- else
-   sk = kmalloc(prot->obj_size, priority);
+ struct sock *sk;

+ sk = sk_prot_alloc(prot, priority);
  if (sk) {
    if (zero_it) {
      memset(sk, 0, prot->obj_size);
@@ -911,10 +931,7 @@ struct sock *sk_alloc(struct net *net, int family, gfp_t priority,
  return sk;

out_free:
- if (slab != NULL)
-   kmem_cache_free(slab, sk);
- else
-   kfree(sk);
+ sk_prot_free(prot, sk);
  return NULL;
}

@@ -940,10 +957,7 @@ void sk_free(struct sock *sk)

  security_sk_free(sk);
  put_net(sk->sk_net);
- if (sk->sk_prot_creator->slab != NULL)
-   kmem_cache_free(sk->sk_prot_creator->slab, sk);
- else
-   kfree(sk);
+ sk_prot_free(sk->sk_prot_creator, sk);
  module_put(owner);
}

--
1.5.3.4

```

---