
Subject: [PATCH] Consolidate creation of kmem caches with "calculated" names
Posted by [Pavel Emelianov](#) on Wed, 17 Oct 2007 13:09:30 GMT

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

Some places in network (like protocol registration and dccp) generate the kmem cache name with `snprintf()` to create caches for several protocols with similar names.

Make the routine that makes this in one place. Possibly, this is better to be put in `mm/sl[uoa]b.c`, but I haven't found any other places in kernel that require such functionality, so put this code (temporary?) in `net/core/util.c`

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

```
diff --git a/include/net/sock.h b/include/net/sock.h
index 453c79d..8040d59 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -601,6 +601,10 @@ struct proto {
extern int proto_register(struct proto *prot, int alloc_slab);
extern void proto_unregister(struct proto *prot);
```

```
+struct kmem_cache *kmem_cache_create_va(int size, unsigned long flags,
+ char *fmt, ...) __attribute__((format (printf, 3, 4)));
+void kmem_cache_destroy_va(struct kmem_cache *);
+
```

```
#ifdef SOCK_REFCNT_DEBUG
static inline void sk_refcnt_debug_inc(struct sock *sk)
{
```

```
diff --git a/net/core/sock.c b/net/core/sock.c
index d45ecdc..65f485a 100644
```

```
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -1777,10 +1777,6 @@ static LIST_HEAD(proto_list);
```

```
int proto_register(struct proto *prot, int alloc_slab)
{
```

```
- char *request_sock_slab_name = NULL;
- char *timewait_sock_slab_name;
- int rc = -ENOBUFS;
-
```

```
if (alloc_slab) {
    prot->slab = kmem_cache_create(prot->name, prot->obj_size, 0,
        SLAB_HWCACHE_ALIGN, NULL);
```

```
@@ -1792,62 +1788,44 @@ int proto_register(struct proto *prot, int alloc_slab)
```

```

}

if (prot->rsk_prot != NULL) {
- static const char mask[] = "request_sock_%s";
-
- request_sock_slab_name = kmalloc(strlen(prot->name) + sizeof(mask) - 1, GFP_KERNEL);
- if (request_sock_slab_name == NULL)
- goto out_free_sock_slab;
-
- sprintf(request_sock_slab_name, mask, prot->name);
- prot->rsk_prot->slab = kmem_cache_create(request_sock_slab_name,
-     prot->rsk_prot->obj_size, 0,
-     SLAB_HWCACHE_ALIGN, NULL);
+ prot->rsk_prot->slab = kmem_cache_create_va(
+     prot->rsk_prot->obj_size,
+     SLAB_HWCACHE_ALIGN,
+     "request_sock_%s", prot->name);

if (prot->rsk_prot->slab == NULL) {
    printk(KERN_CRIT "%s: Can't create request sock SLAB cache!\n",
        prot->name);
- goto out_free_request_sock_slab_name;
+ goto out_rsk;
}
}

if (prot->twsk_prot != NULL) {
- static const char mask[] = "tw_sock_%s";
+ prot->twsk_prot->twsk_slab = kmem_cache_create_va(
+     prot->twsk_prot->twsk_obj_size,
+     SLAB_HWCACHE_ALIGN,
+     "tw_sock_%s", prot->name);

- timewait_sock_slab_name = kmalloc(strlen(prot->name) + sizeof(mask) - 1, GFP_KERNEL);
-
- if (timewait_sock_slab_name == NULL)
- goto out_free_request_sock_slab;
-
- sprintf(timewait_sock_slab_name, mask, prot->name);
- prot->twsk_prot->twsk_slab =
-     kmem_cache_create(timewait_sock_slab_name,
-         prot->twsk_prot->twsk_obj_size,
-         0, SLAB_HWCACHE_ALIGN,
-         NULL);
if (prot->twsk_prot->twsk_slab == NULL)
- goto out_free_timewait_sock_slab_name;
+ goto out_twsk;
}

```

```

}

write_lock(&proto_list_lock);
list_add(&prot->node, &proto_list);
write_unlock(&proto_list_lock);
- rc = 0;
-out:
- return rc;
-out_free_timewait_sock_slab_name:
- kfree(timewait_sock_slab_name);
-out_free_request_sock_slab:
+ return 0;
+
+out_twsk:
+ if (prot->rsk_prot && prot->rsk_prot->slab) {
- kmem_cache_destroy(prot->rsk_prot->slab);
+ kmem_cache_destroy_va(prot->rsk_prot->slab);
+ prot->rsk_prot->slab = NULL;
+ }
-out_free_request_sock_slab_name:
- kfree(request_sock_slab_name);
-out_free_sock_slab:
+out_rsk:
+ kmem_cache_destroy(prot->slab);
+ prot->slab = NULL;
- goto out;
+out:
+ return -ENOBUFS;
}

```

```

EXPORT_SYMBOL(proto_register);
diff --git a/net/core/utls.c b/net/core/utls.c
index 0bf17da..eccc71b 100644
--- a/net/core/utls.c
+++ b/net/core/utls.c
@@ -293,3 +293,41 @@ out:
}

```

```

EXPORT_SYMBOL(in6_pton);
+
+struct kmem_cache *kmem_cache_create_va(int size, unsigned long flags,
+ char *fmt, ...)
+{
+ char *name, str[64];
+ struct kmem_cache *c;
+ va_list args;
+
+ va_start(args, fmt);

```

```

+ vsnprintf(str, sizeof(str), fmt, args);
+ va_end(args);
+
+ name = kstrdup(str, GFP_KERNEL);
+ if (name == NULL)
+ goto err_name;
+
+ c = kmem_cache_create(name, size, 0, flags, NULL);
+ if (c == NULL)
+ goto err_cache;
+
+ return c;
+
+err_cache:
+ kfree(name);
+err_name:
+ return NULL;
+}
+EXPORT_SYMBOL_GPL(kmem_cache_create_va);
+
+void kmem_cache_destroy_va(struct kmem_cache *c)
+{
+ const char *name;
+
+ name = kmem_cache_name(c);
+ kmem_cache_destroy(c);
+ kfree(name);
+}
+EXPORT_SYMBOL_GPL(kmem_cache_destroy_va);
diff --git a/net/dccp/ccid.c b/net/dccp/ccid.c
index c45088b..9d2cc27 100644
--- a/net/dccp/ccid.c
+++ b/net/dccp/ccid.c
@@ -56,51 +56,21 @@ static inline void ccids_read_unlock(void)
#define ccids_read_unlock() do { } while(0)
#endif

-static struct kmem_cache *ccid_kmem_cache_create(int obj_size, const char *fmt,...)
-{
- struct kmem_cache *slab;
- char slab_name_fmt[32], *slab_name;
- va_list args;
-
- va_start(args, fmt);
- vsnprintf(slab_name_fmt, sizeof(slab_name_fmt), fmt, args);
- va_end(args);
-
- slab_name = kstrdup(slab_name_fmt, GFP_KERNEL);

```

```

- if (slab_name == NULL)
- return NULL;
- slab = kmem_cache_create(slab_name, sizeof(struct ccid) + obj_size, 0,
-     SLAB_HWCACHE_ALIGN, NULL);
- if (slab == NULL)
- kfree(slab_name);
- return slab;
-}
-
-static void ccid_kmem_cache_destroy(struct kmem_cache *slab)
-{
- if (slab != NULL) {
-     const char *name = kmem_cache_name(slab);
-
-     kmem_cache_destroy(slab);
-     kfree(name);
- }
-}
-
int ccid_register(struct ccid_operations *ccid_ops)
{
    int err = -ENOBUFFS;

    ccid_ops->ccid_hc_rx_slab =
-     ccid_kmem_cache_create(ccid_ops->ccid_hc_rx_obj_size,
-         "%s_hc_rx_sock",
-         ccid_ops->ccid_name);
+     kmem_cache_create_va(ccid_ops->ccid_hc_rx_obj_size,
+         SLAB_HWCACHE_ALIGN,
+         "%s_hc_rx_sock", ccid_ops->ccid_name);
    if (ccid_ops->ccid_hc_rx_slab == NULL)
        goto out;

    ccid_ops->ccid_hc_tx_slab =
-     ccid_kmem_cache_create(ccid_ops->ccid_hc_tx_obj_size,
-         "%s_hc_tx_sock",
-         ccid_ops->ccid_name);
+     kmem_cache_create_va(ccid_ops->ccid_hc_tx_obj_size,
+         SLAB_HWCACHE_ALIGN,
+         "%s_hc_tx_sock", ccid_ops->ccid_name);
    if (ccid_ops->ccid_hc_tx_slab == NULL)
        goto out_free_rx_slab;

@@ -118,12 +88,13 @@ int ccid_register(struct ccid_operations *ccid_ops)
    ccid_ops->ccid_id, ccid_ops->ccid_name);
out:
    return err;
+

```

```

out_free_tx_slab:
- ccid_kmem_cache_destroy(ccid_ops->ccid_hc_tx_slab);
+ kmem_cache_destroy_va(ccid_ops->ccid_hc_tx_slab);
  ccid_ops->ccid_hc_tx_slab = NULL;
  goto out;
out_free_rx_slab:
- ccid_kmem_cache_destroy(ccid_ops->ccid_hc_rx_slab);
+ kmem_cache_destroy_va(ccid_ops->ccid_hc_rx_slab);
  ccid_ops->ccid_hc_rx_slab = NULL;
  goto out;
}
@@ -136,9 +107,9 @@ int ccid_unregister(struct ccid_operations *ccid_ops)
  ccids[ccid_ops->ccid_id] = NULL;
  ccids_write_unlock();

- ccid_kmem_cache_destroy(ccid_ops->ccid_hc_tx_slab);
+ kmem_cache_destroy_va(ccid_ops->ccid_hc_tx_slab);
  ccid_ops->ccid_hc_tx_slab = NULL;
- ccid_kmem_cache_destroy(ccid_ops->ccid_hc_rx_slab);
+ kmem_cache_destroy_va(ccid_ops->ccid_hc_rx_slab);
  ccid_ops->ccid_hc_rx_slab = NULL;

  pr_info("CCID: Unregistered CCID %d (%s)\n",

```
