
Subject: [PATCH 11/16] Add support for multiple kmem caches for pids

Posted by [Pavel Emelianov](#) on Fri, 06 Jul 2007 08:09:21 GMT

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

Unlike Suka's patches I don't limit the level of pid nesting
creating the caches on demand, depending on the namespace's level.

Each kmem cache is named "pid_<NR>", where <NR> is the level
of pid namespace and thus - the number of virtual pids in it.

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

pid.c | 61 ++-----
1 files changed, 56 insertions(+), 5 deletions(-)

--- ./kernel/pid.c.ve10 2007-07-06 11:04:15.000000000 +0400

+++ ./kernel/pid.c 2007-07-06 11:04:48.000000000 +0400

@@ -32,7 +32,6 @@

#define pid_hashfn(nr) hash_long((unsigned long)nr, pidhash_shift)

static struct hlist_head *pid_hash;

static int pidhash_shift;

-static struct kmem_cache *pid_cachep;

struct pid init_struct_pid = INIT_STRUCT_PID;

int pid_max = PID_MAX_DEFAULT;

@@ -179,11 +178,15 @@ static int next_pidmap(struct pid_namespace

fastcall void put_pid(struct pid *pid)

{

+ struct pid_namespace *ns;

+

if (!pid)

return;

+

+ ns = pid->numbers[0].ns;

if ((atomic_read(&pid->count) == 1) ||
atomic_dec_and_test(&pid->count))

- kmem_cache_free(pid_cachep, pid);

+ kmem_cache_free(ns->pid_cachep, pid);

}

EXPORT_SYMBOL_GPL(put_pid);

@@ -212,7 +215,7 @@ struct pid *alloc_pid(struct pid_namespace

enum pid_type type;

int nr = -1;

```
- pid = kmem_cache_alloc(pid_cachep, GFP_KERNEL);
+ pid = kmem_cache_alloc(init_pid_ns.pid_cachep, GFP_KERNEL);
  if (!pid)
    goto out;
```

```
@@ -233,7 +236,7 @@ out:
    return pid;
```

```
out_free:
- kmem_cache_free(pid_cachep, pid);
+ kmem_cache_free(init_pid_ns.pid_cachep, pid);
  pid = NULL;
  goto out;
}
```

```
@@ -378,6 +381,52 @@ struct pid *find_ge_pid(int nr, struct p
}
EXPORT_SYMBOL_GPL(find_get_pid);
```

```
+struct pid_cache {
+ int level;
+ char name[16];
+ struct kmem_cache *cachep;
+ struct list_head lh;
+};
+
+static LIST_HEAD(pid_caches);
+static DEFINE_MUTEX(pid_cache_mutex);
+
+static struct kmem_cache *create_pid_cachep(int level)
+{
+ struct pid_cache *pc;
+ struct kmem_cache *cachep = NULL;
+
+ mutex_lock(&pid_cache_mutex);
+ list_for_each_entry(pc, &pid_caches, lh)
+ if (pc->level == level) {
+   cachep = pc->cachep;
+   goto out;
+ }
+
+ pc = kzalloc(sizeof(struct pid_cache), GFP_KERNEL);
+ if (pc == NULL)
+   goto out;
+
+ snprintf(pc->name, sizeof(pc->name), "pid_%d", level);
+ cachep = kmem_cache_create(pc->name,
+   sizeof(struct pid) + level * sizeof(struct pid_number),
+   0, SLAB_HWCACHE_ALIGN, NULL, NULL);
```

```

+ if (cachep == NULL)
+ goto out_free;
+
+ pc->cachep = cachep;
+ pc->level = level;
+ list_add(&pc->lh, &pid_caches);
+ pc = NULL;
+
+out_free:
+ if (pc != NULL)
+ kfree(pc);
+out:
+ mutex_unlock(&pid_cache_mutex);
+ return cachep;
+}
+
struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *old_ns)
{
    BUG_ON(!old_ns);
@@ -425,5 +474,7 @@ void __init pidmap_init(void)
    set_bit(0, init_pid_ns.pidmap[0].page);
    atomic_dec(&init_pid_ns.pidmap[0].nr_free);

- pid_cachep = KMEM_CACHE(pid, SLAB_PANIC);
+ init_pid_ns.pid_cachep = create_pid_cachep(0);
+ if (init_pid_ns.pid_cachep == NULL)
+ panic("Can't create pid cachep");
}

```

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