Subject: [PATCH 3/3] Dynamic kmem cache allocator for pid namespaces
Posted by Pavel Emelianov on Tue, 10 Jul 2007 14:28:10 GMT
View Forum Message <> Reply to Message

Add kmem_cache to pid_namespace to allocate pids from.

Since booth implementations expand the struct pid to carry
more numerical values each namespace should have separate
cache to store pids of different sizes.

Each kmem cache is names "pid_<NR>", where <NR> is the number
of numerical ids on the pid. Different namespaces with same
level of nesting will have same caches.

This patch has two FIXMEs that are to be fixed after we reach
the consensus about the struct pid itself.

The first one is that the namespace to free the pid from in
free_pid() must be taken from pid. Now the init_pid_ns is
used.

The second FIXME is about the cache allocation. When we do know
how long the object will be then we'll have to calculate this
size in create_pid_cachep. Right now the sizeof(struct pid)
value is used.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>
Acked-by: Cedric Le Goater <clg@fr.ibm.com>
Acked-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

```
diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
index ddb9a4c..27cfad3 100644
--- a/include/linux/pid_namespace.h
+++ b/include/linux/pid_namespace.h
@@ -20,6 +20,7 @@ struct pid_namespace {
  struct pidmap pidmap[PIDMAP_ENTRIES];
  int last_pid;
  struct task_struct *child_reaper;
+ struct kmem_cache_t *pid_cachep;
 };

 extern struct pid_namespace init_pid_ns;
diff --git a/kernel/pid.c b/kernel/pid.c
index c6e3f9f..d3f0f91 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
```

```
@@ -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;
@@ -176,11 +175,16 @@ static int next_pidmap(struct pid_namesp

 fastcall void put_pid(struct pid *pid)
 {
+ struct pid_namespace *ns;
+
  if (!pid)
   return;
+
+ /* FIXME - this must be the namespace this pid lives in */
+ ns = &init_pid_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);

@@ -208,12 +212,14 @@ struct pid *alloc_pid(void)
  struct pid *pid;
  enum pid_type type;
  int nr = -1;
+ struct pid_namespace *ns;

- pid = kmem_cache_alloc(pid_cachep, GFP_KERNEL);
+ ns = current->nsproxy->pid_ns;
+ pid = kmem_cache_alloc(ns->pid_cachep, GFP_KERNEL);
  if (!pid)
   goto out;

- nr = alloc_pidmap(current->nsproxy->pid_ns);
+ nr = alloc_pidmap(ns);
  if (nr < 0)
   goto out_free;

@@ -230,7 +236,7 @@ out:
  return pid;

 out_free:
- kmem_cache_free(pid_cachep, pid);
```

```
+ kmem_cache_free(ns->pid_cachep, pid);
  pid = NULL;
  goto out;
 }
@@ -365,6 +371,56 @@ struct pid *find_ge_pid(int nr)
 }
 EXPORT_SYMBOL_GPL(find_get_pid);

+struct pid_cache {
+ int nr_ids;
+ char name[16];
+ struct kmem_cache *cachep;
+ struct list_head list;
+};
+
+static LIST_HEAD(pid_caches_lh);
+static DEFINE_MUTEX(pid_caches_mutex);
+
+/*
+ * creates the kmem cache to allocate pids from.
+ * @nr_ids: the number of numerical ids this pid will have to carry
+ */
+
+static struct kmem_cache *create_pid_cachep(int nr_ids)
+{
+ struct pid_cache *pcache;
+ struct kmem_cache *cachep;
+
+ mutex_lock(&pid_caches_mutex);
+ list_for_each_entry (pcache, &pid_caches_lh, list)
+  if (pcache->nr_ids == nr_ids)
+   goto out;
+
+ pcache = kmalloc(sizeof(struct pid_cache), GFP_KERNEL);
+ if (pcache == NULL)
+  goto err_alloc;
+
+ snprintf(pcache->name, sizeof(pcache->name), "pid_%d", nr_ids);
+ cachep = kmem_cache_create(pcache->name,
+   /* FIXME add numerical ids here */
+   sizeof(struct pid), 0, SLAB_HWCACHE_ALIGN, NULL, NULL);
+ if (cachep == NULL)
+  goto err_cachep;
+
+ pcache->nr_ids = nr_ids;
+ pcache->cachep = cachep;
+ list_add(&pcache->list, &pid_caches_lh);
+out:
```

```
+ mutex_unlock(&pid_caches_mutex);
+ return pcache->cachep;
+
+err_cachep:
+ kfree(pcache);
+err_alloc:
+ mutex_unlock(&pid_caches_mutex);
+ return NULL;
+}
+
 struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *old_ns)
 {
  BUG_ON(!old_ns);
@@ -412,5 +468,7 @@ 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(1);
+ if (init_pid_ns.pid_cachep == NULL)
+  panic("Can't create pid_1 cachep\n");
 }
```

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