
Subject: [RFC][PATCH 1/3] Setup the kmem container
Posted by [Pavel Emelianov](#) on Thu, 13 Sep 2007 09:13:11 GMT
[View Forum Message](#) <> [Reply to Message](#)

[PATCH 1/3] Setup the kmem container

Attach the controller to the containers. This can work with the SLUB allocator only. However, if we need I can port this on SLAB (and even SLOB ;)).

This setup is simple and stupid.

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

```
include/linux/container_subsys.h | 6 ++
init/Kconfig                     | 6 ++
mm/slub.c                       | 110 +++++
3 files changed, 122 insertions(+)
```

```
diff --git a/include/linux/container_subsys.h b/include/linux/container_subsys.h
index 81d11c2..9dd90d9 100644
--- a/include/linux/container_subsys.h
+++ b/include/linux/container_subsys.h
@@ -36,3 +36,9 @@ SUBSYS(mem_container)
#endif
```

```
/* */
+
+#ifdef CONFIG_CONTAINER_KMEM
+SUBSYS(kmem)
+#endif
+
+/* */
diff --git a/init/Kconfig b/init/Kconfig
index 58559ea..d499f15 100644
--- a/init/Kconfig
+++ b/init/Kconfig
@@ -353,6 +353,12 @@ config CONTAINER_MEM_CONT
    Provides a memory controller that manages both page cache and
    RSS memory.

+config CONTAINER_KMEM
+ bool "Kernel memory controller for containers"
+ depends on CONTAINERS && RESOURCE_COUNTERS && SLUB
+ help
+   Provides a kernel memory usage control for containers
```

```

+
config PROC_PID_CPUSET
    bool "Include legacy /proc/<pid>/cpuset file"
    depends on CPUSETS
diff --git a/mm/slub.c b/mm/slub.c
index 1802645..16da4d4 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -21,6 +21,12 @@
#include <linux/ctype.h>
#include <linux/kallsyms.h>

#ifdef CONFIG_CONTAINER_KMEM
#include <linux/container.h>
#include <linux/res_counter.h>
#include <linux/err.h>
#endif
+
/*
 * Lock order:
 * 1. slab_lock(page)
@@ -4011,3 +4016,107 @@ static int __init slab_sysfs_init(void)

__initcall(slab_sysfs_init);
#endif
+
#ifdef CONFIG_CONTAINER_KMEM
+struct kmem_container {
+ struct container_subsys_state css;
+ struct res_counter res;
+};
+
+static inline
+struct kmem_container *css_to_kmem(struct container_subsys_state *css)
+{
+ return container_of(css, struct kmem_container, css);
+}
+
+static inline
+struct kmem_container *container_to_kmem(struct container *cont)
+{
+ return css_to_kmem(container_subsys_state(cont, kmem_subsys_id));
+}
+
+static inline
+struct kmem_container *task_kmem_container(struct task_struct *tsk)
+{
+ return css_to_kmem(task_subsys_state(tsk, kmem_subsys_id));

```

```

+}
+
+/*
+ * containers interface
+ */
+
+static struct kmem_container init_kmem_container;
+
+static struct container_subsys_state *kmem_create(struct container_subsys *ss,
+ struct container *container)
+{
+ struct kmem_container *mem;
+
+ if (unlikely((container->parent) == NULL))
+ mem = &init_kmem_container;
+ else
+ mem = kzalloc(sizeof(struct kmem_container), GFP_KERNEL);
+
+ if (mem == NULL)
+ return ERR_PTR(-ENOMEM);
+
+ res_counter_init(&mem->res);
+ return &mem->css;
+
+}
+
+static void kmem_destroy(struct container_subsys *ss,
+ struct container *container)
+{
+ kfree(container_to_kmem(container));
+}
+
+static ssize_t kmem_container_read(struct container *cont, struct cftype *cft,
+ struct file *file, char __user *userbuf, size_t nbytes,
+ loff_t *ppos)
+{
+ return res_counter_read(&container_to_kmem(cont)->res,
+ cft->private, userbuf, nbytes, ppos);
+}
+
+static ssize_t kmem_container_write(struct container *cont, struct cftype *cft,
+ struct file *file, const char __user *userbuf,
+ size_t nbytes, loff_t *ppos)
+{
+ return res_counter_write(&container_to_kmem(cont)->res,
+ cft->private, userbuf, nbytes, ppos);
+}
+

```

```

+static struct cftype kmem_files[] = {
+ {
+ .name = "usage",
+ .private = RES_USAGE,
+ .read = kmem_container_read,
+ },
+ {
+ .name = "limit",
+ .private = RES_LIMIT,
+ .write = kmem_container_write,
+ .read = kmem_container_read,
+ },
+ {
+ .name = "failcnt",
+ .private = RES_FAILCNT,
+ .read = kmem_container_read,
+ },
+};
+
+static int kmem_populate(struct container_subsys *ss, struct container *cnt)
+{
+ return container_add_files(cnt, ss, kmem_files, ARRAY_SIZE(kmem_files));
+}
+
+struct container_subsys kmem_subsys = {
+ .name = "kmem",
+ .create = kmem_create,
+ .destroy = kmem_destroy,
+ .populate = kmem_populate,
+ .subsys_id = kmem_subsys_id,
+ .early_init = 1,
+};
+#endif

```
