Subject: [PATCH v4 02/25] provide a common place for initcall processing in kmem_cache
Posted by Glauber Costa on Mon, 18 Jun 2012 10:27:55 GMT
View Forum Message <> Reply to Message

Both SLAB and SLUB depend on some initialization to happen when the
system is already booted, with all subsystems working. This is done
by issuing an initcall that does the final initialization.

This patch moves that to slab_common.c, while creating an empty
placeholder for the SLOB.

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
CC: David Rientjes <rientjes@google.com>
---
 mm/slab.c        |    5 ++---
 mm/slab.h        |    1 +
 mm/slab_common.c |    5 +++++
 mm/slob.c        |    5 +++++
 mm/slub.c        |    4 +---
 5 files changed, 14 insertions(+), 6 deletions(-)

diff --git a/mm/slab.c b/mm/slab.c
index 020605f..e174e50 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -853,7 +853,7 @@ static void __cpuinit start_cpu_timer(int cpu)
  struct delayed_work *reap_work = &per_cpu(slab_reap_work, cpu);

  /*
-  * When this gets called from do_initcalls via cpucache_init(),
+  * When this gets called from do_initcalls via __kmem_cache_initcall(),
   * init_workqueues() has already run, so keventd will be setup
   * at that time.
   */
@@ -1666,7 +1666,7 @@ void __init kmem_cache_init_late(void)
  */
 }

-static int __init cpucache_init(void)
+int __init __kmem_cache_initcall(void)
 {
  int cpu;

@@ -1677,7 +1677,6 @@ static int __init cpucache_init(void)
  start_cpu_timer(cpu);

```
  return 0;
 }
-__initcall(cpucache_init);

 static noinline void
 slab_out_of_memory(struct kmem_cache *cachep, gfp_t gfpflags, int nodeid)
diff --git a/mm/slab.h b/mm/slab.h
index b44a8cc..19e17c7 100644
--- a/mm/slab.h
+++ b/mm/slab.h
@@ -37,6 +37,7 @@ unsigned long calculate_alignment(unsigned long flags,

 /* Functions provided by the slab allocators */
 int __kmem_cache_create(struct kmem_cache *s);
+int __kmem_cache_initcall(void);

 #ifdef CONFIG_SLUB
 struct kmem_cache *__kmem_cache_alias(const char *name, size_t size,
diff --git a/mm/slab_common.c b/mm/slab_common.c
index b5def07..15db694ac 100644
--- a/mm/slab_common.c
+++ b/mm/slab_common.c
@@ -203,3 +203,8 @@ int slab_is_available(void)
  return slab_state >= UP;
 }

+static int __init kmem_cache_initcall(void)
+{
+ return __kmem_cache_initcall();
+}
+__initcall(kmem_cache_initcall);
diff --git a/mm/slob.c b/mm/slob.c
index 507589d..61b1845 100644
--- a/mm/slob.c
+++ b/mm/slob.c
@@ -610,3 +610,8 @@ void __init kmem_cache_init(void)
 void __init kmem_cache_init_late(void)
 {
 }
+
+int __init kmem_cache_initcall(void)
+{
+ return 0;
+}
diff --git a/mm/slub.c b/mm/slub.c
index 7fc3499..d9d4d5a 100644
--- a/mm/slub.c
+++ b/mm/slub.c
```

```
@@ -5309,7 +5309,7 @@ static int sysfs_slab_alias(struct kmem_cache *s, const char *name)
  return 0;
 }

-static int __init slab_sysfs_init(void)
+int __init __kmem_cache_initcall(void)
 {
  struct kmem_cache *s;
  int err;
@@ -5347,8 +5347,6 @@ static int __init slab_sysfs_init(void)
  resiliency_test();
  return 0;
 }
-
-__initcall(slab_sysfs_init);
 #endif /* CONFIG_SYSFS */

 /*
--
1.7.10.2
```