Subject: [PATCH v3 04/16] provide a common place for initcall processing in kmem_cache
Posted by Glauber Costa on Tue, 18 Sep 2012 14:11:58 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 | 2 ++
 mm/slab_common.c | 6 ++++++
 mm/slob.c | 5 +++++
 mm/slub.c | 4 +---
 5 files changed, 16 insertions(+), 6 deletions(-)

diff --git a/mm/slab.c b/mm/slab.c
index ccf496c..3bac667 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -894,7 +894,7 @@ 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.
   */
@@ -1822,7 +1822,7 @@ void __init kmem_cache_init_late(void)
  */
 }

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

@@ -1836,7 +1836,6 @@ static int __init cpucache_init(void)
  slab_state = FULL;

```
  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 7deeb44..7a2698b 100644
--- a/mm/slab.h
+++ b/mm/slab.h
@@ -47,4 +47,6 @@ static inline struct kmem_cache *__kmem_cache_alias(const char *name,
size_t siz

 int __kmem_cache_shutdown(struct kmem_cache *);

+int __kmem_cache_initcall(void);
+
 #endif
diff --git a/mm/slab_common.c b/mm/slab_common.c
index 9c21725..eddbb8a 100644
--- a/mm/slab_common.c
+++ b/mm/slab_common.c
@@ -189,3 +189,9 @@ 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 3edfeaa..ad91d67 100644
--- a/mm/slob.c
+++ b/mm/slob.c
@@ -622,3 +622,8 @@ void __init kmem_cache_init_late(void)
 {
  slab_state = FULL;
 }
+
+int __init __kmem_cache_initcall(void)
+{
+ return 0;
+}
diff --git a/mm/slub.c b/mm/slub.c
index 09a91d0..7ac46c6 100644
--- a/mm/slub.c
```

```
+++ b/mm/slub.c
@@ -5332,7 +5332,7 @@ 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;
@@ -5370,8 +5370,6 @@ static int __init slab_sysfs_init(void)
  resiliency_test();
  return 0;
 }
-
-__initcall(slab_sysfs_init);
 #endif /* CONFIG_SYSFS */

 /*
--
1.7.11.4
```