Subject: [PATCH] provide a common place for initcall processing in kmem_cache
Posted by Glauber Costa on Mon, 23 Jul 2012 08:33:28 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>

Pekka: This is the last slab change I need that is independent of memcg. All
further changes are either not in the slab files, or done to accomodate the
memcg changes. The reason I need this bit, is to register the memcg indexes,
that can only be done after the caches are up (memcg_register_cache()).

```
 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 1fcf3ac..2cf636a 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -863,7 +863,7 @@ 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.
   */
@@ -1665,7 +1665,7 @@ void __init kmem_cache_init_late(void)
   */
 }

-static int __init cpucache_init(void)
+int __init __kmem_cache_initcall(void)
```

```
 {
  int cpu;

@@ -1679,7 +1679,6 @@ 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 db7848c..1d17049 100644
--- a/mm/slab.h
+++ b/mm/slab.h
@@ -30,4 +30,6 @@ extern struct list_head slab_caches;
 struct kmem_cache *__kmem_cache_create(const char *name, size_t size,
  size_t align, unsigned long flags, void (*ctor)(void *));

+int __kmem_cache_initcall(void);
+
 #endif
diff --git a/mm/slab_common.c b/mm/slab_common.c
index 12637ce..af50ba0 100644
--- a/mm/slab_common.c
+++ b/mm/slab_common.c
@@ -116,3 +116,9 @@ 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 45d4ca7..6c2b5d1 100644
--- a/mm/slob.c
+++ b/mm/slob.c
@@ -628,3 +628,8 @@ 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 e517d43..07b1c3e 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -5348,7 +5348,7 @@ 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;
@@ -5386,8 +5386,6 @@ static int __init slab_sysfs_init(void)
  resiliency_test();
  return 0;
 }
-
-__initcall(slab_sysfs_init);
 #endif /* CONFIG_SYSFS */

 /*
--
1.7.10.4
```

Subject: Re: [PATCH] provide a common place for initcall processing in kmem_cache
Posted by Christoph Lameter on Tue, 24 Jul 2012 15:43:15 GMT
View Forum Message <> Reply to Message

On Mon, 23 Jul 2012, Glauber Costa wrote:

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

Acked-by: Christoph Lameter <cl@linux.com>
