Subject: [RFC 3/4] memcg: do not call page_cgroup_init at system_boot
Posted by Glauber Costa on Tue, 25 Sep 2012 08:52:52 GMT
View Forum Message <> Reply to Message

If we are not using memcg, there is no reason why we should allocate
this structure, that will be a memory waste at best. We can do better
at least in the sparsemem case, and allocate it when the first cgroup
is requested. It should now not panic on failure, and we have to handle
this right.

flatmem case is a bit more complicated, so that one is left out for
the moment.

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Michal Hocko <mhocko@suse.cz>
CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
CC: Johannes Weiner <hannes@cmpxchg.org>
CC: Mel Gorman <mgorman@suse.de>
CC: Andrew Morton <akpm@linux-foundation.org>
---
 include/linux/page_cgroup.h | 20 ++++++++++++++-----
 init/main.c | 1 -
 mm/memcontrol.c | 5 +++++
 mm/page_cgroup.c | 40 ++--------
 4 files changed, 52 insertions(+), 14 deletions(-)

diff --git a/include/linux/page_cgroup.h b/include/linux/page_cgroup.h
index 777a524..46dd0e7 100644
--- a/include/linux/page_cgroup.h
+++ b/include/linux/page_cgroup.h
@@ -27,16 +27,21 @@ struct page_cgroup {
  struct mem_cgroup *mem_cgroup;
 };

-void __meminit pgdat_page_cgroup_init(struct pglist_data *pgdat);
+void pgdat_page_cgroup_init(struct pglist_data *pgdat);

 #ifdef CONFIG_SPARSEMEM
 static inline void __init page_cgroup_init_flatmem(void)
 {
 }
-extern void __init page_cgroup_init(void);
+extern int page_cgroup_init(void);
+extern void page_cgroup_destroy(void);
 #else
 void __init page_cgroup_init_flatmem(void);
-static inline void __init page_cgroup_init(void)
+static inline int page_cgroup_init(void)

```
+{
+}
+
+static inline void page_cgroup_destroy(void)
 {
 }
 #endif
@@ -85,7 +90,7 @@ static inline void unlock_page_cgroup(struct page_cgroup *pc)
 #else /* CONFIG_MEMCG */
 struct page_cgroup;

-static inline void __meminit pgdat_page_cgroup_init(struct pglist_data *pgdat)
+static inline void pgdat_page_cgroup_init(struct pglist_data *pgdat)
 {
 }

@@ -94,7 +99,12 @@ static inline struct page_cgroup *lookup_page_cgroup(struct page *page)
  return NULL;
 }

-static inline void page_cgroup_init(void)
+static inline int page_cgroup_init(void)
+{
+ return 0;
+}
+
+static inline void page_cgroup_destroy(void)
 {
 }

diff --git a/init/main.c b/init/main.c
index 09cf339..9c48d1c 100644
--- a/init/main.c
+++ b/init/main.c
@@ -588,7 +588,6 @@ asmlinkage void __init start_kernel(void)
   initrd_start = 0;
  }
 #endif
- page_cgroup_init();
  debug_objects_mem_init();
  kmemleak_init();
  setup_per_cpu_pageset();
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index a2c88c4..f8115f0 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -5709,6 +5709,10 @@ mem_cgroup_create(struct cgroup *cont)
  }
```

```
   hotcpu_notifier(memcg_cpu_hotplug_callback, 0);
  } else {
+  /* FIXME: Need to clean this up properly if fails */
+  if (!page_cgroup_init())
+   goto free_out;
+
   parent = mem_cgroup_from_cont(cont->parent);
   memcg->use_hierarchy = parent->use_hierarchy;
   memcg->oom_kill_disable = parent->oom_kill_disable;
@@ -5789,6 +5793,7 @@ static void mem_cgroup_destroy(struct cgroup *cont)
  kmem_cgroup_destroy(memcg);

  static_key_slow_dec(&memcg_in_use_key);
+ page_cgroup_destroy();
  mem_cgroup_put(memcg);
 }

diff --git a/mm/page_cgroup.c b/mm/page_cgroup.c
index 5699e9f..ee5738f 100644
--- a/mm/page_cgroup.c
+++ b/mm/page_cgroup.c
@@ -16,7 +16,7 @@ static unsigned long total_usage;
 #if !defined(CONFIG_SPARSEMEM)


-void __meminit pgdat_page_cgroup_init(struct pglist_data *pgdat)
+void pgdat_page_cgroup_init(struct pglist_data *pgdat)
 {
  pgdat->node_page_cgroup = NULL;
 }
@@ -42,7 +42,7 @@ struct page_cgroup *lookup_page_cgroup(struct page *page)
  return base + offset;
 }

-static int __init alloc_node_page_cgroup(int nid)
+static int alloc_node_page_cgroup(int nid)
 {
  struct page_cgroup *base;
  unsigned long table_size;
@@ -105,7 +105,7 @@ struct page_cgroup *lookup_page_cgroup(struct page *page)
  return section->page_cgroup + pfn;
 }

-static void *__meminit alloc_page_cgroup(size_t size, int nid)
+static void *alloc_page_cgroup(size_t size, int nid)
 {
  gfp_t flags = GFP_KERNEL | __GFP_ZERO | __GFP_NOWARN;
  void *addr = NULL;
```

```
@@ -124,7 +124,7 @@ static void *__meminit alloc_page_cgroup(size_t size, int nid)
  return addr;
 }

-static int __meminit init_section_page_cgroup(unsigned long pfn, int nid)
+static int init_section_page_cgroup(unsigned long pfn, int nid)
 {
  struct mem_section *section;
  struct page_cgroup *base;
@@ -263,7 +263,9 @@ static int __meminit page_cgroup_callback(struct notifier_block *self,

 #endif

-void __init page_cgroup_init(void)
+static atomic_t page_cgroup_initialized = ATOMIC_INIT(0);
+
+int page_cgroup_init(void)
 {
  unsigned long pfn;
  int nid;
@@ -271,6 +273,12 @@ void __init page_cgroup_init(void)
  if (mem_cgroup_subsys_disabled())
   return;

+ if (atomic_add_return(1, &page_cgroup_initialized) != 1)
+  return 0;
+
+ /* We can arrive here multiple times, if memcgs come and go. */
+ total_usage = 0;
+
  for_each_node_state(nid, N_HIGH_MEMORY) {
   unsigned long start_pfn, end_pfn;

@@ -306,16 +314,32 @@ void __init page_cgroup_init(void)
  return;
 oom:
  printk(KERN_CRIT "try 'cgroup_disable=memory' boot option\n");
- panic("Out of memory");
+ return -ENOMEM;
 }

-void __meminit pgdat_page_cgroup_init(struct pglist_data *pgdat)
+void pgdat_page_cgroup_init(struct pglist_data *pgdat)
 {
  return;
 }

-#endif
```

```
+void page_cgroup_destroy(void)
+{
+ int nid;

+ if (atomic_sub_return(1, &page_cgroup_initialized))
+  return;
+
+ for_each_node_state(nid, N_HIGH_MEMORY) {
+  unsigned long start, end, pfn;
+
+  start = node_start_pfn(nid);
+  end = node_end_pfn(nid);
+
+  for (pfn = start; pfn < end; pfn += PAGES_PER_SECTION)
+   __free_page_cgroup(pfn);
+ }
+}
+#endif

 #ifdef CONFIG_MEMCG_SWAP

--
1.7.11.4
```