Subject: [PATCH v3 15/28] slub: always get the cache from its page in kfree
Posted by Glauber Costa on Fri, 25 May 2012 13:03:35 GMT
View Forum Message <> Reply to Message

struct page already have this information. If we start chaining
caches, this information will always be more trustworthy than
whatever is passed into the function

A parent pointer is added to the slub structure, so we can make sure
the freeing comes from either the right slab, or from its rightful
parent.

[v3: added parent testing with VM_BUG_ON]

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
---
 include/linux/slab.h | 3 +++
 include/linux/slub_def.h | 18 ++++++++++++++++++
 mm/slub.c | 7 ++++++-
 3 files changed, 27 insertions(+), 1 deletions(-)

diff --git a/include/linux/slab.h b/include/linux/slab.h
index e73ef71..724c143 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -164,6 +164,9 @@ struct mem_cgroup_cache_params {
  size_t orig_align;

 #endif
+#ifdef CONFIG_DEBUG_VM
+ struct kmem_cache *parent;
+#endif
 };
 #endif

diff --git a/include/linux/slub_def.h b/include/linux/slub_def.h
index 5f5e942..f822ca2 100644
--- a/include/linux/slub_def.h
+++ b/include/linux/slub_def.h
@@ -115,6 +115,24 @@ struct kmem_cache {
  struct kmem_cache_node *node[MAX_NUMNODES];
 };

+static inline void slab_set_parent(struct kmem_cache *s,
+   struct kmem_cache *parent)
+{

```
+#if defined(CONFIG_CGROUP_MEM_RES_CTLR_KMEM) && defined(CONFIG_DEBUG_VM)
+ s->memcg_params.parent = parent;
+#endif
+}
+
+static inline bool slab_is_parent(struct kmem_cache *s,
+      struct kmem_cache *candidate)
+{
+#if defined(CONFIG_CGROUP_MEM_RES_CTLR_KMEM) && defined(CONFIG_DEBUG_VM)
+ return candidate == s->memcg_params.parent;
+#else
+ return false;
+#endif
+}
+
 /*
  * Kmalloc subsystem.
  */
diff --git a/mm/slub.c b/mm/slub.c
index 0eb9e72..640872f 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -2598,10 +2598,14 @@ redo:
 void kmem_cache_free(struct kmem_cache *s, void *x)
 {
  struct page *page;
+ bool slab_match;

  page = virt_to_head_page(x);

- slab_free(s, page, x, _RET_IP_);
+ slab_match = (page->slab == s) | slab_is_parent(page->slab, s);
+ VM_BUG_ON(!slab_match);
+
+ slab_free(page->slab, page, x, _RET_IP_);

  trace_kmem_cache_free(_RET_IP_, x);
 }
@@ -4027,6 +4031,7 @@ struct kmem_cache *kmem_cache_dup(struct mem_cgroup *memcg,
   down_write(&slub_lock);
   s->refcount++;
   up_write(&slub_lock);
+  slab_set_parent(new, s);
  }
  /* slub internals is expected to have held a copy of it */
  kfree(name);
--
1.7.7.6
```