Subject: [PATCH 1/4] Add notification about some major slab events
Posted by Pavel Emelianov on Mon, 17 Sep 2007 12:26:23 GMT
View Forum Message <> Reply to Message

According to Christoph, there are already multiple people who
want to control slab allocations and track memory for various
reasons. So this is an introduction of such a hooks.

The selected method of notification is srcu notifier blocks.
This is selected because the "call" path, i.e. when the
notification is done, is lockless and at the same time
notification handlers can sleep. Neither regular nor atomic
notifiers provide such facilities.

The events tracked are:
1. allocation of an object
2. freeing of an onbject
3. allocation of a new page for objects
4. freeing this page

More events can be added on demand.

The kmem cache marked with SLAB_NOTIFY flag will cause all the
events above to generate notifications. By default no caches
come with this flag.

To preserve the fast-paths and keep the stack from growing
the checks for the flag are made in a separate inline functions
and the actual notification is done in noinline ones.

Hopefully, this looks close to how Christoph sees it :)

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
---

 include/linux/slab.h     |    1
 include/linux/slub_def.h |   16 +++++++
 mm/slub.c                |  105 ++++++++++++++++++++++++++++++++++++++++++++++++++++++-
 3 files changed, 121 insertions(+), 1 deletion(-)

diff --git a/include/linux/slab.h b/include/linux/slab.h
index 3a5bad3..a3bd620 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -28,6 +28,7 @@
 #define SLAB_DESTROY_BY_RCU 0x00080000UL /* Defer freeing slabs to RCU */
 #define SLAB_MEM_SPREAD  0x00100000UL /* Spread some memory over cpuset */
```

```
 #define SLAB_TRACE  0x00200000UL /* Trace allocations and frees */
+#define SLAB_NOTIFY  0x00400000UL /* Notify major events */

 /* The following flags affect the page allocator grouping pages by mobility */
 #define SLAB_RECLAIM_ACCOUNT 0x00020000UL  /* Objects are reclaimable */
diff --git a/include/linux/slub_def.h b/include/linux/slub_def.h
index d65159d..547777e 100644
--- a/include/linux/slub_def.h
+++ b/include/linux/slub_def.h
@@ -200,4 +202,20 @@ static __always_inline void *kmalloc_nod
 }
 #endif

+struct slub_notify_struct {
+ struct kmem_cache *cachep;
+ void *objp;
+ gfp_t gfp;
+};
+
+enum {
+ SLUB_ALLOC,
+ SLUB_FREE,
+ SLUB_NEWPAGE,
+ SLUB_FREEPAGE,
+};
+
+int slub_register_notifier(struct notifier_block *nb);
+void slub_unregister_notifier(struct notifier_block *nb);
+
 #endif /* _LINUX_SLUB_DEF_H */
diff --git a/mm/slub.c b/mm/slub.c
index 1802645..bfb7c21 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -1013,6 +1013,91 @@ static inline void add_full(struct kmem_
 static inline void kmem_cache_open_debug_check(struct kmem_cache *s) {}
 #define slub_debug 0
 #endif
+
+/*
+ * notifiers
+ */
+
+static struct srcu_notifier_head slub_nb;
+
+static noinline
+int __slub_alloc_notify(int cmd_alloc, int cmd_free, struct kmem_cache *cachep,
+  void *obj, gfp_t gfp)
```

```
+{
+ int ret, called;
+ struct slub_notify_struct arg;
+
+ arg.cachep = cachep;
+ arg.objp = obj;
+ arg.gfp = gfp;
+
+ ret = __srcu_notifier_call_chain(&slub_nb, cmd_alloc, &arg,
+   -1, &called);
+ ret = notifier_to_errno(ret);
+
+ if (ret < 0)
+  __srcu_notifier_call_chain(&slub_nb, cmd_free, &arg,
+    called, NULL);
+
+ return ret;
+}
+
+static noinline
+void __slub_free_notify(int cmd, struct kmem_cache *cachep, void *obj)
+{
+ struct slub_notify_struct arg;
+
+ arg.cachep = cachep;
+ arg.objp = obj;
+ arg.gfp = 0;
+
+ srcu_notifier_call_chain(&slub_nb, cmd, &arg);
+}
+
+int slub_register_notifier(struct notifier_block *nb)
+{
+ return srcu_notifier_chain_register(&slub_nb, nb);
+}
+
+void slub_unregister_notifier(struct notifier_block *nb)
+{
+ srcu_notifier_chain_unregister(&slub_nb, nb);
+}
+
+/*
+ * fastpath hooks
+ */
+
+static inline
+int slub_alloc_notify(struct kmem_cache *cachep, void *obj, gfp_t gfp)
+{
```

```
+ return (unlikely(cachep->flags & SLAB_NOTIFY)) ?
+  __slub_alloc_notify(SLUB_ALLOC, SLUB_FREE,
+    cachep, obj, gfp) : 0;
+}
+
+static inline
+void slub_free_notify(struct kmem_cache *cachep, void *obj)
+{
+ if (unlikely(cachep->flags & SLAB_NOTIFY))
+  __slub_free_notify(SLUB_FREE, cachep, obj);
+}
+
+static inline
+int slub_newpage_notify(struct kmem_cache *cachep, struct page *pg, gfp_t gfp)
+{
+ return (unlikely(cachep->flags & SLAB_NOTIFY)) ?
+  __slub_alloc_notify(SLUB_NEWPAGE, SLUB_FREEPAGE,
+    cachep, pg, gfp) : 0;
+}
+
+static inline
+void slub_freepage_notify(struct kmem_cache *cachep, struct page *pg)
+{
+ if (unlikely(cachep->flags & SLAB_NOTIFY))
+  __slub_free_notify(SLUB_FREEPAGE, cachep, pg);
+}
+
 /*
  * Slab allocation and freeing
  */
@@ -1036,7 +1121,10 @@ static struct page *allocate_slab(struct
   page = alloc_pages_node(node, flags, s->order);

  if (!page)
-  return NULL;
+  goto out;
+
+ if (slub_newpage_notify(s, page, flags) < 0)
+  goto out_free;

  mod_zone_page_state(page_zone(page),
   (s->flags & SLAB_RECLAIM_ACCOUNT) ?
@@ -1044,6 +1132,11 @@ static struct page *allocate_slab(struct
   pages);

  return page;
+
+out_free:
```

```
+ __free_pages(page, s->order);
+out:
+ return NULL;
 }

 static void setup_object(struct kmem_cache *s, struct page *page,
@@ -1136,6 +1229,8 @@ static void rcu_free_slab(struct rcu_hea

 static void free_slab(struct kmem_cache *s, struct page *page)
 {
+ slub_freepage_notify(s, page);
+
  if (unlikely(s->flags & SLAB_DESTROY_BY_RCU)) {
   /*
    * RCU free overloads the RCU head over the LRU
@@ -1555,6 +1650,11 @@ static void __always_inline *slab_alloc(
  }
  local_irq_restore(flags);

+ if (object && slub_alloc_notify(s, object, gfpflags) < 0) {
+  kmem_cache_free(s, object);
+  return NULL;
+ }
+
  if (unlikely((gfpflags & __GFP_ZERO) && object))
   memset(object, 0, c->objsize);

@@ -1651,6 +1751,8 @@ static void __always_inline slab_free(st
  unsigned long flags;
  struct kmem_cache_cpu *c;

+ slub_free_notify(s, x);
+
  local_irq_save(flags);
  debug_check_no_locks_freed(object, s->objsize);
  c = get_cpu_slab(s, smp_processor_id());
@@ -2764,6 +2874,7 @@ void __init kmem_cache_init(void)
  kmem_size = sizeof(struct kmem_cache);
 #endif

+ srcu_init_notifier_head(&slub_nb);

  printk(KERN_INFO "SLUB: Genslabs=%d, HWalign=%d, Order=%d-%d, MinObjects=%d,"
   " CPUs=%d, Nodes=%d\n",
```