Subject: [PATCH 1/4] Add notification about some major slab events
Posted by Pavel Emelianov on Fri, 21 Sep 2007 09:17:17 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.

Currently, functions that are to call the notifiers are empty
and marked as "weak". Thus, if there's only _one_ listener
to these events, it can happily link with the vmlinux and
handle the events with more than 10% of performance saved.

The events tracked are:
1. allocation of an object;
2. freeing of an object;
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.

The events are generated on slow paths only and as soon as the
cache is marked as SLAB_NOTIFY, it will always use them for
allocation.

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

```
diff --git a/include/linux/slab.h b/include/linux/slab.h
index f3a8eec..68d8e65 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/mm/slub.c b/mm/slub.c
index ac4f157..b5af598 100644
--- a/mm/slub.c
```

```
+++ b/mm/slub.c
@@ -1040,6 +1040,29 @@ static inline unsigned long kmem_cache_f
 }
 #define slub_debug 0
 #endif
+
+int __attribute__ ((weak))
+slub_alloc_notify(struct kmem_cache *cachep, void *obj, gfp_t gfp)
+{
+ return 0;
+}
+
+void __attribute__ ((weak))
+slub_free_notify(struct kmem_cache *cachep, void *obj)
+{
+}
+
+int __attribute__ ((weak))
+slub_newpage_notify(struct kmem_cache *cachep, struct page *pg, gfp_t gfp)
+{
+ return 0;
+}
+
+void __attribute__ ((weak))
+slub_freepage_notify(struct kmem_cache *cachep, struct page *pg)
+{
+}
+
 /*
  * Slab allocation and freeing
  */
@@ -1063,7 +1162,11 @@ static struct page *allocate_slab(struct
   page = alloc_pages_node(node, flags, s->order);

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

  mod_zone_page_state(page_zone(page),
   (s->flags & SLAB_RECLAIM_ACCOUNT) ?
@@ -1071,6 +1174,11 @@ 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,
@@ -1158,6 +1266,9 @@ static void rcu_free_slab(struct rcu_hea

 static void free_slab(struct kmem_cache *s, struct page *page)
 {
+ if (s->flags & SLAB_NOTIFY)
+  slub_freepage_notify(s, page);
+
  if (unlikely(s->flags & SLAB_DESTROY_BY_RCU)) {
   /*
    * RCU free overloads the RCU head over the LRU
@@ -1486,7 +1597,7 @@ load_freelist:
  object = c->page->freelist;
  if (unlikely(!object))
   goto another_slab;
- if (unlikely(SlabDebug(c->page)))
+ if (unlikely(SlabDebug(c->page)) || (s->flags & SLAB_NOTIFY))
   goto debug;

  object = c->page->freelist;
@@ -1545,12 +1656,20 @@ new_slab:
  return NULL;
 debug:
  object = c->page->freelist;
- if (!alloc_debug_processing(s, c->page, object, addr))
+ if (SlabDebug(c->page) &&
+   !alloc_debug_processing(s, c->page, object, addr))
   goto another_slab;

+ if ((s->flags & SLAB_NOTIFY) &&
+   slub_alloc_notify(s, object, gfpflags) < 0) {
+  object = NULL;
+  goto out;
+ }
+
  c->page->inuse++;
  c->page->freelist = object[c->offset];
  c->node = -1;
+out:
  slab_unlock(c->page);
  return object;
```

```
 }
@@ -1620,7 +1739,7 @@ static void __slab_free(struct kmem_cach

  slab_lock(page);

- if (unlikely(SlabDebug(page)))
+ if (unlikely(SlabDebug(page)) || (s->flags & SLAB_NOTIFY))
    goto debug;
 checks_ok:
  prior = object[offset] = page->freelist;
@@ -1657,8 +1776,12 @@ slab_empty:
  return;

 debug:
- if (!free_debug_processing(s, page, x, addr))
+ if (SlabDebug(page) && !free_debug_processing(s, page, x, addr))
    goto out_unlock;
+
+ if (s->flags & SLAB_NOTIFY)
+  slub_free_notify(s, x);
+
  goto checks_ok;
 }
```