
Subject: [PATCH 1/5] Add notification about some major slab events

Posted by [Pavel Emelianov](#) on Tue, 25 Sep 2007 14:18:00 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/include/linux/slub_def.h b/include/linux/slub_def.h
index 40801e7..8cfd9ff 100644
--- a/include/linux/slub_def.h
```

```

+++ b/include/linux/slub_def.h
@@ -200,4 +203,22 @@ static __always_inline void *kmalloc_nod
}
#endif

+struct slub_notify_struct {
+ struct kmem_cache *cachep;
+ void *objp;
+ gfp_t gfp;
+};
+
+enum {
+ SLUB_ON,
+ SLUB_OFF,
+ 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 31d04a3..e066a0e 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -1040,6 +1040,43 @@ static inline unsigned long kmem_cache_f
}
#define slub_debug 0
#endif

+
+/*
+ * notifiers
+ */
+
+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)
+{
+}
+
+int __attribute__((weak)) slub_on_notify(struct kmem_cache *cachep)
+{
+ return 0;
+}
+
+int __attribute__((weak)) slub_off_notify(struct kmem_cache *cachep)
+{
+ return 0;
+}
+
+/*
+ * Slab allocation and freeing
+ */
@@ -1063,7 +1184,11 @@ 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 +1196,11 @@ 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,
@@ -1103,7 +1233,7 @@ static struct page *new_slab(struct kmem
    page->slab = s;
    page->flags |= 1 << PG_slab;
    if (s->flags & (SLAB_DEBUG_FREE | SLAB_RED_ZONE | SLAB_POISON |
-   SLAB_STORE_USER | SLAB_TRACE))
+   SLAB_STORE_USER | SLAB_TRACE | SLAB_NOTIFY))
        SetSlabDebug(page);

    start = page_address(page);
@@ -1158,6 +1288,9 @@ static void rcu_free_slab(struct rcu_he

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
@@ -1548,9 +1681,16 @@ debug:
        if (!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;
}
@@ -1659,6 +1799,10 @@ slab_empty:
debug:
    if (!free_debug_processing(s, page, x, addr))
        goto out_unlock;
+
+ if (s->flags & SLAB_NOTIFY)
+   slub_free_notify(s, x);
+
    goto checks_ok;
}

```
