
Subject: [PATCH 8/8] Per-container pages reclamation
Posted by [Pavel Emelianov](#) on Wed, 30 May 2007 15:38:01 GMT
[View Forum Message](#) <> [Reply to Message](#)

Implement `try_to_free_pages_in_container()` to free the pages in container that has run out of memory.

The `scan_control->isolate_pages()` function is set to `isolate_pages_in_container()` that isolates the container pages only. The exported `__isolate_lru_page()` call makes things look simpler than in the previous version.

Includes fix from Balbir Singh <balbir@in.ibm.com>

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

```
diff -upr linux-2.6.22-rc2-mm1.orig/mm/rss_container.c
linux-2.6.22-rc2-mm1-0/mm/rss_container.c
--- linux-2.6.22-rc2-mm1.orig/mm/rss_container.c 2007-05-30 16:16:58.000000000 +0400
+++ linux-2.6.22-rc2-mm1-0/mm/rss_container.c 2007-05-30 16:13:09.000000000 +0400
@@ -135,6 +135,76 @@
     spin_unlock(&rss->res.lock);
 }
```

```
+void container_rss_move_lists(struct page *pg, bool active)
+{
+ struct rss_container *rss;
+ struct page_container *pc;
+
+ if (!page_mapped(pg))
+ return;
+
+ pc = page_container(pg);
+ if (pc == NULL)
+ return;
+
+ rss = pc->cnt;
+
+ spin_lock(&rss->res.lock);
+ if (active)
+ list_move(&pc->list, &rss->active_list);
+ else
+ list_move(&pc->list, &rss->inactive_list);
+ spin_unlock(&rss->res.lock);
+}
+
```

```

+static unsigned long isolate_container_pages(unsigned long nr_to_scan,
+ struct list_head *src, struct list_head *dst,
+ unsigned long *scanned, struct zone *zone, int mode)
+{
+ unsigned long nr_taken = 0;
+ struct page *page;
+ struct page_container *pc;
+ unsigned long scan;
+ LIST_HEAD(pc_list);
+
+ for (scan = 0; scan < nr_to_scan && !list_empty(src); scan++) {
+ pc = list_entry(src->prev, struct page_container, list);
+ page = pc->page;
+ if (page_zone(page) != zone)
+ continue;
+
+ list_move(&pc->list, &pc_list);
+
+ if (__isolate_lru_page(page, mode) == 0) {
+ list_move(&page->lru, dst);
+ nr_taken++;
+ }
+ }
+
+ list_splice(&pc_list, src);
+
+ *scanned = scan;
+ return nr_taken;
+}
+
+unsigned long isolate_pages_in_container(unsigned long nr_to_scan,
+ struct list_head *dst, unsigned long *scanned,
+ int order, int mode, struct zone *zone,
+ struct rss_container *rss, int active)
+{
+ unsigned long ret;
+
+ spin_lock(&rss->res.lock);
+ if (active)
+ ret = isolate_container_pages(nr_to_scan, &rss->active_list,
+ dst, scanned, zone, mode);
+ else
+ ret = isolate_container_pages(nr_to_scan, &rss->inactive_list,
+ dst, scanned, zone, mode);
+ spin_unlock(&rss->res.lock);
+ return ret;
+}
+

```

```

void container_rss_add(struct page_container *pc)
{
    struct page *pg;
diff -upr linux-2.6.22-rc2-mm1.orig/mm/swap.c linux-2.6.22-rc2-mm1-0/mm/swap.c
--- linux-2.6.22-rc2-mm1.orig/mm/swap.c 2007-05-30 12:32:36.000000000 +0400
+++ linux-2.6.22-rc2-mm1-0/mm/swap.c 2007-05-30 16:13:09.000000000 +0400
@@ -31,6 +31,7 @@
#include <linux/cpu.h>
#include <linux/notifier.h>
#include <linux/init.h>
+#include <linux/rss_container.h>

/* How many pages do we try to swap or page in/out together? */
int page_cluster;
@@ -148,6 +149,7 @@ void fastcall activate_page(struct page
    SetPageActive(page);
    add_page_to_active_list(zone, page);
    __count_vm_event(PGACTIVATE);
+ container_rss_move_lists(page, 1);
}
spin_unlock_irq(&zone->lru_lock);
}
diff -upr linux-2.6.22-rc2-mm1.orig/include/linux/rss_container.h
linux-2.6.22-rc2-mm1-0/include/linux/rss_container.h
--- linux-2.6.22-rc2-mm1.orig/include/linux/rss_container.h 2007-05-30 16:16:58.000000000
+0400
+++ linux-2.6.22-rc2-mm1-0/include/linux/rss_container.h 2007-05-30 16:13:09.000000000 +0400
@@ -24,6 +24,11 @@
void mm_init_container(struct mm_struct *mm, struct task_struct *tsk);
void mm_free_container(struct mm_struct *mm);

+void container_rss_move_lists(struct page *pg, bool active);
+unsigned long isolate_pages_in_container(unsigned long nr_to_scan,
+ struct list_head *dst, unsigned long *scanned,
+ int order, int mode, struct zone *zone,
+ struct rss_container *, int active);
static inline void init_page_container(struct page *page)
{
    page_container(page) = NULL;
@@ -64,5 +68,7 @@
{
}

+#define isolate_container_pages(nr, dst, scanned, rss, act, zone) ({ BUG(); 0; })
+#define container_rss_move_lists(pg, active) do { } while (0)
#endif
#endif
diff -upr linux-2.6.22-rc2-mm1.orig/mm/vmscan.c linux-2.6.22-rc2-mm1-0/mm/vmscan.c

```

```

--- linux-2.6.22-rc2-mm1.orig/mm/vmscan.c 2007-05-30 12:32:36.000000000 +0400
+++ linux-2.6.22-rc2-mm1-0/mm/vmscan.c 2007-05-30 16:13:09.000000000 +0400
@@ -983,6 +1005,7 @@ force_reclaim_mapped:
    ClearPageActive(page);

    list_move(&page->lru, &zone->inactive_list);
+ container_rss_move_lists(page, 0);
    pgmoved++;
    if (!pagevec_add(&pvec, page)) {
        __mod_zone_page_state(zone, NR_INACTIVE, pgmoved);
@@ -1011,6 +1034,7 @@ force_reclaim_mapped:
    SetPageLRU(page);
    VM_BUG_ON(!PageActive(page));
    list_move(&page->lru, &zone->active_list);
+ container_rss_move_lists(page, 1);
    pgmoved++;
    if (!pagevec_add(&pvec, page)) {
        __mod_zone_page_state(zone, NR_ACTIVE, pgmoved);
@@ -1230,6 +1248,36 @@ out:
    return ret;
}

#ifdef CONFIG_RSS_CONTAINER
+unsigned long try_to_free_pages_in_container(struct rss_container *cnt)
+{
+ struct scan_control sc = {
+ .gfp_mask = GFP_KERNEL,
+ .may_wrotepage = 1,
+ .swap_cluster_max = 1,
+ .may_swap = 1,
+ .swappiness = vm_swappiness,
+ .order = 0, /* in this case we wanted one page only */
+ .cnt = cnt,
+ .isolate_pages = isolate_pages_in_container,
+ };
+ int node;
+ struct zone **zones;
+
+ for_each_online_node(node) {
+ #ifdef CONFIG_HIGHMEM
+ zones = NODE_DATA(node)->node_zonelist[ZONE_HIGHMEM].zones;
+ if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
+ return 1;
+ #endif
+ zones = NODE_DATA(node)->node_zonelist[ZONE_NORMAL].zones;
+ if (do_try_to_free_pages(zones, sc.gfp_mask, &sc))
+ return 1;
+ }
}

```

```
+ return 0;
+}
+ #endif
+
+ unsigned long try_to_free_pages(struct zone **zones, int order, gfp_t gfp_mask)
+ {
+     struct scan_control sc = {
```
