
Subject: [PATCH 8/8] Per-container pages reclamation

Posted by [xemul](#) on Mon, 09 Apr 2007 12:59:50 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 isolates the container pages only.

```
diff -upr linux-2.6.20.orig/include/linux/swap.h linux-2.6.20-2/include/linux/swap.h
--- linux-2.6.20.orig/include/linux/swap.h 2007-03-06 19:09:50.000000000 +0300
+++ linux-2.6.20-2/include/linux/swap.h 2007-04-09 11:26:06.000000000 +0400
@@ -187,6 +187,9 @@ extern void swap_setup(void);
```

```
/* linux/mm/vmscan.c */
extern unsigned long try_to_free_pages(struct zone **, gfp_t);
+
+struct rss_container;
+extern unsigned long try_to_free_pages_in_container(struct rss_container *);
extern unsigned long shrink_all_memory(unsigned long nr_pages);
extern int vm_swappiness;
extern int remove_mapping(struct address_space *mapping, struct page *page);
diff -upr linux-2.6.20.orig/mm/vmscan.c linux-2.6.20-2/mm/vmscan.c
--- linux-2.6.20.orig/mm/vmscan.c 2007-03-06 19:09:50.000000000 +0300
+++ linux-2.6.20-2/mm/vmscan.c 2007-04-09 11:26:06.000000000 +0400
@@ -872,6 +872,7 @@ force_reclaim_mapped:
    ClearPageActive(page);
```

```
    list_move(&page->lru, &zone->inactive_list);
+ container_rss_move_lists(page, 0);
    pgmoved++;
    if (!pagevec_add(&pvec, page)) {
        zone->nr_inactive += pgmoved;
@@ -900,6 +901,7 @@ 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)) {
        zone->nr_active += pgmoved;
@@ -1110,6 +1112,35 @@ 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_writepage = 1,
+ .swap_cluster_max = 1,
+ .may_swap = 1,
+ .swappiness = vm_swappiness,
+ .cnt = cnt,
+ .isolate_pages = isolate_pages_in_container,
+ };
+ int node;
+ struct zone **zones;
+
+ for_each_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;
+}
+
+
+ unsigned long try_to_free_pages(struct zone **zones, gfp_t gfp_mask)
+ {
+     struct scan_control sc = {
diff -upr linux-2.6.20.orig/mm/swap.c linux-2.6.20-2/mm/swap.c
--- linux-2.6.20.orig/mm/swap.c 2007-03-06 19:09:50.000000000 +0300
+++ linux-2.6.20-2/mm/swap.c 2007-04-09 11:26:06.000000000 +0400
@@ -30,6 +30,7 @@
#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;
@@ -147,6 +148,7 @@ 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-2.6.20.orig/mm/rss_container.c linux-2.6.20-2/mm/rss_container.c
--- linux-2.6.20.orig/mm/rss_container.c 2007-04-09 11:26:12.000000000 +0400
+++ linux-2.6.20-2/mm/rss_container.c 2007-04-09 11:26:06.000000000 +0400
@@ -96,6 +96,78 @@
    kfree(pc);
}
```

```
+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)
+{
+ 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 (PageLRU(page)) {
+   if (likely(get_page_unless_zero(page))) {
+     ClearPageLRU(page);
+     nr_taken++;
+     list_move(&page->lru, dst);
+   }
+ }
+ }
+ }
+
+ 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,
+ 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);
+ else
+   ret = isolate_container_pages(nr_to_scan, &rss->inactive_list,
+   dst, scanned, zone);
+ spin_unlock(&rss->res.lock);
+ return ret;
+}
+
+void container_rss_add(struct page_container *pc)
+{
+   struct page *pg;
diff -upr linux-2.6.20.orig/include/linux/rss_container.h linux-2.6.20-2/include/linux/rss_container.h
--- linux-2.6.20.orig/include/linux/rss_container.h 2007-04-09 11:26:12.000000000 +0400
+++ linux-2.6.20-2/include/linux/rss_container.h 2007-04-09 11:26:06.000000000 +0400
@@ -24,6 +24,10 @@
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,
+ struct zone *zone, struct rss_container *, int active);
#else

```

```
static inline int container_rss_prepare(struct page *pg,
    struct vm_area_struct *vma, struct page_container **pc)
@@ -52,5 +56,7 @@
{
}

+#define isolate_container_pages(nr, dst, scanned, rss, act, zone) ({ BUG(); 0;})
+#define container_rss_move_lists(pg, active) do { } while (0)
#endif
#endif
```
