Subject: [PATCH 2/3][RFC] Containers: Pagecache controller accounting
Posted by Vaidyanathan Srinivas on Mon, 05 Mar 2007 14:52:39 GMT
View Forum Message <> Reply to Message

The accounting framework works by adding a container pointer in
address_space structure. Each page in pagecache belongs to a
radix tree within the address_space structure corresponding to the inode.

In order to charge the container for pagecache usage, the corresponding
address_space is obtained from struct page which holds the container pointer.
This framework avoids any additional pointers in struct page.

additions and deletions from pagecache are hooked to charge and uncharge
the corresponding container.

Signed-off-by: Vaidyanathan Srinivasan <svaidy@linux.vnet.ibm.com>
---
include/linux/fs.h | 4 ++++
mm/filemap.c | 8 ++++++++
2 files changed, 12 insertions(+)

```
--- linux-2.6.20.orig/include/linux/fs.h
+++ linux-2.6.20/include/linux/fs.h
@@ -447,6 +447,10 @@ struct address_space {
  spinlock_t  private_lock; /* for use by the address_space */
  struct list_head private_list; /* ditto */
  struct address_space *assoc_mapping; /* ditto */
+#ifdef CONFIG_CONTAINER_PAGECACHE_ACCT
+ struct container *container;  /* Charge page to the right container
+        using page->mapping */
+#endif
 } __attribute__((aligned(sizeof(long))));
  /*
   * On most architectures that alignment is already the case; but
--- linux-2.6.20.orig/mm/filemap.c
+++ linux-2.6.20/mm/filemap.c
@@ -30,6 +30,7 @@
 #include <linux/security.h>
 #include <linux/syscalls.h>
 #include <linux/cpuset.h>
+#include <linux/pagecache_acct.h>
 #include "filemap.h"
 #include "internal.h"

@@ -117,6 +118,8 @@ void __remove_from_page_cache(struct pag
  struct address_space *mapping = page->mapping;

  radix_tree_delete(&mapping->page_tree, page->index);
```

```
+ /* Uncharge before the mapping is gone */
+ pagecache_acct_uncharge(page);
  page->mapping = NULL;
  mapping->nrpages--;
  __dec_zone_page_state(page, NR_FILE_PAGES);
@@ -451,6 +454,11 @@ int add_to_page_cache(struct page *page,
    __inc_zone_page_state(page, NR_FILE_PAGES);
   }
   write_unlock_irq(&mapping->tree_lock);
+  /* Unlock before charge, because we may reclaim this inline */
+  if (!error) {
+   pagecache_acct_init_page_ptr(page);
+   pagecache_acct_charge(page);
+  }
   radix_tree_preload_end();
  }
  return error;

--
```