Subject: [PATCH 3/5] page_cgroup: make page tracking available for blkio
Posted by Andrea Righi on Tue, 22 Feb 2011 17:12:54 GMT
View Forum Message <> Reply to Message

The page_cgroup infrastructure, currently available only for the memory
cgroup controller, can be used to store the owner of each page and
opportunely track the writeback IO. This information is encoded in
the upper 16-bits of the page_cgroup->flags.

A owner can be identified using a generic ID number and the following
interfaces are provided to store a retrieve this information:

```
  unsigned long page_cgroup_get_owner(struct page *page);
  int page_cgroup_set_owner(struct page *page, unsigned long id);
  int page_cgroup_copy_owner(struct page *npage, struct page *opage);
```

The blkio.throttle controller can use the cgroup css_id() as the owner's
ID number.

Signed-off-by: Andrea Righi <arighi@develer.com>

```
---
 block/Kconfig                |    2 +
 block/blk-cgroup.c           |    6 ++
 include/linux/memcontrol.h   |    6 ++
 include/linux/mmzone.h       |    4 +-
 include/linux/page_cgroup.h  |   33 ++++++++++-
 init/Kconfig                 |    4 +
 mm/Makefile                  |    3 +-
 mm/memcontrol.c              |    6 ++
 mm/page_cgroup.c             |  129 +++++++++++++++++++++++++++++++++++++++----
 9 files changed, 176 insertions(+), 17 deletions(-)

diff --git a/block/Kconfig b/block/Kconfig
index 60be1e0..1351ea8 100644
--- a/block/Kconfig
+++ b/block/Kconfig
@@ -80,6 +80,8 @@ config BLK_DEV_INTEGRITY
 config BLK_DEV_THROTTLING
  bool "Block layer bio throttling support"
  depends on BLK_CGROUP=y && EXPERIMENTAL
+ select MM_OWNER
+ select PAGE_TRACKING
  default n
  ---help---
  Block layer bio throttling support. It can be used to limit
diff --git a/block/blk-cgroup.c b/block/blk-cgroup.c
index f283ae1..5c57f0a 100644
--- a/block/blk-cgroup.c
```

```
+++ b/block/blk-cgroup.c
@@ -107,6 +107,12 @@ blkio_policy_search_node(const struct blkio_cgroup *blkcg, dev_t dev,
  return NULL;
 }

+bool blkio_cgroup_disabled(void)
+{
+ return blkio_subsys.disabled ? true : false;
+}
+EXPORT_SYMBOL_GPL(blkio_cgroup_disabled);
+
 struct blkio_cgroup *task_to_blkio_cgroup(struct task_struct *task)
 {
  return container_of(task_subsys_state(task, blkio_subsys_id),
diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index f512e18..a8a7cf0 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -49,6 +49,8 @@ extern unsigned long mem_cgroup_isolate_pages(unsigned long
nr_to_scan,
  * (Of course, if memcg does memory allocation in future, GFP_KERNEL is sane.)
  */

+extern void __init_mem_page_cgroup(struct page_cgroup *pc);
+
 extern int mem_cgroup_newpage_charge(struct page *page, struct mm_struct *mm,
     gfp_t gfp_mask);
 /* for swap handling */
@@ -153,6 +155,10 @@ void mem_cgroup_split_huge_fixup(struct page *head, struct page
*tail);
 #else /* CONFIG_CGROUP_MEM_RES_CTLR */
 struct mem_cgroup;

+static inline void __init_mem_page_cgroup(struct page_cgroup *pc)
+{
+}
+
 static inline int mem_cgroup_newpage_charge(struct page *page,
      struct mm_struct *mm, gfp_t gfp_mask)
 {
diff --git a/include/linux/mmzone.h b/include/linux/mmzone.h
index 02ecb01..30a5938 100644
--- a/include/linux/mmzone.h
+++ b/include/linux/mmzone.h
@@ -615,7 +615,7 @@ typedef struct pglist_data {
  int nr_zones;
 #ifdef CONFIG_FLAT_NODE_MEM_MAP /* means !SPARSEMEM */
  struct page *node_mem_map;
```

```
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
+#ifdef CONFIG_PAGE_TRACKING
  struct page_cgroup *node_page_cgroup;
 #endif
 #endif
@@ -975,7 +975,7 @@ struct mem_section {

  /* See declaration of similar field in struct zone */
  unsigned long *pageblock_flags;
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
+#ifdef CONFIG_PAGE_TRACKING
  /*
   * If !SPARSEMEM, pgdat doesn't have page_cgroup pointer. We use
   * section. (see memcontrol.h/page_cgroup.h about this.)
diff --git a/include/linux/page_cgroup.h b/include/linux/page_cgroup.h
index 6d6cb7a..cdd7728 100644
--- a/include/linux/page_cgroup.h
+++ b/include/linux/page_cgroup.h
@@ -1,7 +1,7 @@
 #ifndef __LINUX_PAGE_CGROUP_H
 #define __LINUX_PAGE_CGROUP_H

-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
+#ifdef CONFIG_PAGE_TRACKING
 #include <linux/bit_spinlock.h>
 /*
  * Page Cgroup can be considered as an extended mem_map.
@@ -12,11 +12,38 @@
  */
 struct page_cgroup {
  unsigned long flags;
- struct mem_cgroup *mem_cgroup;
  struct page *page;
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR
+ struct mem_cgroup *mem_cgroup;
  struct list_head lru;  /* per cgroup LRU list */
+#endif
 };

+/*
+ * use lower 16 bits for flags and reserve the rest for the page tracking id
+ */
+#define PAGE_TRACKING_ID_SHIFT (16)
+#define PAGE_TRACKING_ID_BITS \
+  (8 * sizeof(unsigned long) - PAGE_TRACKING_ID_SHIFT)
+
+/* NOTE: must be called with page_cgroup() lock held */
+static inline unsigned long page_cgroup_get_id(struct page_cgroup *pc)
```

```
+{
+ return pc->flags >> PAGE_TRACKING_ID_SHIFT;
+}
+
+/* NOTE: must be called with page_cgroup() lock held */
+static inline void page_cgroup_set_id(struct page_cgroup *pc, unsigned long id)
+{
+ WARN_ON(id >= (1UL << PAGE_TRACKING_ID_BITS));
+ pc->flags &= (1UL << PAGE_TRACKING_ID_SHIFT) - 1;
+ pc->flags |= (unsigned long)(id << PAGE_TRACKING_ID_SHIFT);
+}
+
+unsigned long page_cgroup_get_owner(struct page *page);
+int page_cgroup_set_owner(struct page *page, unsigned long id);
+int page_cgroup_copy_owner(struct page *npage, struct page *opage);
+
 void __meminit pgdat_page_cgroup_init(struct pglist_data *pgdat);

 #ifdef CONFIG_SPARSEMEM
@@ -132,7 +159,7 @@ static inline void move_unlock_page_cgroup(struct page_cgroup *pc,
  local_irq_restore(*flags);
 }

-#else /* CONFIG_CGROUP_MEM_RES_CTLR */
+#else /* CONFIG_PAGE_TRACKING */
 struct page_cgroup;

 static inline void __meminit pgdat_page_cgroup_init(struct pglist_data *pgdat)
diff --git a/init/Kconfig b/init/Kconfig
index be788c0..712a00a 100644
--- a/init/Kconfig
+++ b/init/Kconfig
@@ -633,6 +633,7 @@ config CGROUP_MEM_RES_CTLR
  bool "Memory Resource Controller for Control Groups"
  depends on RESOURCE_COUNTERS
  select MM_OWNER
+ select PAGE_TRACKING
  help
    Provides a memory resource controller that manages both anonymous
    memory and page cache. (See Documentation/cgroups/memory.txt)
@@ -813,6 +814,9 @@ config SCHED_AUTOGROUP
 config MM_OWNER
  bool

+config PAGE_TRACKING
+ bool
+
 config SYSFS_DEPRECATED
```

```
  bool "enable deprecated sysfs features to support old userspace tools"
  depends on SYSFS
diff --git a/mm/Makefile b/mm/Makefile
index 2b1b575..85448cc 100644
--- a/mm/Makefile
+++ b/mm/Makefile
@@ -38,7 +38,8 @@ obj-$(CONFIG_FS_XIP) += filemap_xip.o
 obj-$(CONFIG_MIGRATION) += migrate.o
 obj-$(CONFIG_QUICKLIST) += quicklist.o
 obj-$(CONFIG_TRANSPARENT_HUGEPAGE) += huge_memory.o
-obj-$(CONFIG_CGROUP_MEM_RES_CTLR) += memcontrol.o page_cgroup.o
+obj-$(CONFIG_CGROUP_MEM_RES_CTLR) += memcontrol.o
+obj-$(CONFIG_PAGE_TRACKING) += page_cgroup.o
 obj-$(CONFIG_MEMORY_FAILURE) += memory-failure.o
 obj-$(CONFIG_HWPOISON_INJECT) += hwpoison-inject.o
 obj-$(CONFIG_DEBUG_KMEMLEAK) += kmemleak.o
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index da53a25..1f72c2b 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -5056,6 +5056,12 @@ struct cgroup_subsys mem_cgroup_subsys = {
  .use_id = 1,
 };

+void __meminit __init_mem_page_cgroup(struct page_cgroup *pc)
+{
+ pc->mem_cgroup = NULL;
+ INIT_LIST_HEAD(&pc->lru);
+}
+
 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_SWAP
 static int __init enable_swap_account(char *s)
 {
diff --git a/mm/page_cgroup.c b/mm/page_cgroup.c
index 5bffada..79214ae 100644
--- a/mm/page_cgroup.c
+++ b/mm/page_cgroup.c
@@ -2,6 +2,7 @@
 #include <linux/mmzone.h>
 #include <linux/bootmem.h>
 #include <linux/bit_spinlock.h>
+#include <linux/blk-cgroup.h>
 #include <linux/page_cgroup.h>
 #include <linux/hash.h>
 #include <linux/slab.h>
@@ -15,9 +16,8 @@ static void __meminit
 __init_page_cgroup(struct page_cgroup *pc, unsigned long pfn)
 {
```

```
  pc->flags = 0;
- pc->mem_cgroup = NULL;
  pc->page = pfn_to_page(pfn);
- INIT_LIST_HEAD(&pc->lru);
+ __init_mem_page_cgroup(pc);
 }
 static unsigned long total_usage;

@@ -75,7 +75,7 @@ void __init page_cgroup_init_flatmem(void)

  int nid, fail;

- if (mem_cgroup_disabled())
+ if (mem_cgroup_disabled() && blkio_cgroup_disabled())
    return;

  for_each_online_node(nid)  {
@@ -84,12 +84,13 @@ void __init page_cgroup_init_flatmem(void)
     goto fail;
  }
  printk(KERN_INFO "allocated %ld bytes of page_cgroup\n", total_usage);
- printk(KERN_INFO "please try 'cgroup_disable=memory' option if you"
- " don't want memory cgroups\n");
+ printk(KERN_INFO
+  "try cgroup_disable=memory,blkio option if you don't want\n");
  return;
 fail:
  printk(KERN_CRIT "allocation of page_cgroup failed.\n");
- printk(KERN_CRIT "please try 'cgroup_disable=memory' boot option\n");
+ printk(KERN_CRIT
+  "try cgroup_disable=memory,blkio boot option\n");
  panic("Out of memory");
 }

@@ -258,7 +259,7 @@ void __init page_cgroup_init(void)
  unsigned long pfn;
  int fail = 0;

- if (mem_cgroup_disabled())
+ if (mem_cgroup_disabled() && blkio_cgroup_disabled())
    return;

  for (pfn = 0; !fail && pfn < max_pfn; pfn += PAGES_PER_SECTION) {
@@ -267,14 +268,15 @@ void __init page_cgroup_init(void)
    fail = init_section_page_cgroup(pfn);
  }
  if (fail) {
-  printk(KERN_CRIT "try 'cgroup_disable=memory' boot option\n");
```

```
+  printk(KERN_CRIT
+   "try cgroup_disable=memory,blkio boot option\n");
   panic("Out of memory");
  } else {
   hotplug_memory_notifier(page_cgroup_callback, 0);
  }
  printk(KERN_INFO "allocated %ld bytes of page_cgroup\n", total_usage);
- printk(KERN_INFO "please try 'cgroup_disable=memory' option if you don't"
- " want memory cgroups\n");
+ printk(KERN_INFO
+  "try cgroup_disable=memory,blkio option if you don't want\n");
 }

 void __meminit pgdat_page_cgroup_init(struct pglist_data *pgdat)
@@ -282,8 +284,113 @@ void __meminit pgdat_page_cgroup_init(struct pglist_data *pgdat)
  return;
 }

-#endif
+#endif /* !defined(CONFIG_SPARSEMEM) */
+
+/**
+ * page_cgroup_get_owner() - get the owner ID of a page
+ * @page: the page we want to find the owner
+ *
+ * Returns the owner ID of the page, 0 means that the owner cannot be
+ * retrieved.
+ **/
+unsigned long page_cgroup_get_owner(struct page *page)
+{
+ struct page_cgroup *pc;
+ unsigned long ret;
+
+ pc = lookup_page_cgroup(page);
+ if (unlikely(!pc))
+  return 0;
+
+ lock_page_cgroup(pc);
+ ret = page_cgroup_get_id(pc);
+ unlock_page_cgroup(pc);
+ return ret;
+}
+
+/**
+ * page_cgroup_set_owner() - set the owner ID of a page
+ * @page: the page we want to tag
+ * @id:  the ID number that will be associated to page
+ *
```

```
+ * Returns 0 if the owner is correctly associated to the page. Returns a
+ * negative value in case of failure.
+ **/
+int page_cgroup_set_owner(struct page *page, unsigned long id)
+{
+ struct page_cgroup *pc;
+
+ pc = lookup_page_cgroup(page);
+ if (unlikely(!pc))
+  return -ENOENT;
+
+ lock_page_cgroup(pc);
+ page_cgroup_set_id(pc, id);
+ unlock_page_cgroup(pc);
+ return 0;
+}
+
+/*
+ * double_page_cgroup_lock() - safely lock two page cgroups
+ *
+ * The double_page_cgroup_lock() code uses the address of the page cgroup to be
+ * sure to acquire the locks always in the same order and avoid AB-BA deadlock.
+ */
+static void
+double_page_cgroup_lock(struct page_cgroup *pc1, struct page_cgroup *pc2)
+{
+ if (pc1 == pc2) {
+  lock_page_cgroup(pc1);
+ } else {
+  if (pc1 < pc2) {
+   lock_page_cgroup(pc1);
+   lock_page_cgroup(pc2);
+  } else {
+   lock_page_cgroup(pc2);
+   lock_page_cgroup(pc1);
+  }
+ }
+}

+/*
+ * double_unlock_page_cgroup() - safely unlock two page cgroups
+ */
+static void
+double_page_cgroup_unlock(struct page_cgroup *pc1, struct page_cgroup *pc2)
+{
+ unlock_page_cgroup(pc1);
+ if (pc1 != pc2)
+  unlock_page_cgroup(pc2);
```

```
+}
+
+/**
+ * page_cgroup_copy_owner() - copy the owner ID of a page into another page
+ * @npage: the page where we want to copy the owner
+ * @opage: the page from which we want to copy the ID
+ *
+ * Returns 0 if the owner is correctly associated to npage. Returns a negative
+ * value in case of failure.
+ **/
+int page_cgroup_copy_owner(struct page *npage, struct page *opage)
+{
+ struct page_cgroup *npc, *opc;
+ int id;
+
+ npc = lookup_page_cgroup(npage);
+ if (unlikely(!npc))
+  return -ENOENT;
+ opc = lookup_page_cgroup(opage);
+ if (unlikely(!opc))
+  return -ENOENT;
+ if (npc == opc)
+  return 0;
+ double_page_cgroup_lock(npc, opc);
+ id = page_cgroup_get_id(opc);
+ page_cgroup_set_id(npc, id);
+ double_page_cgroup_unlock(npc, opc);
+
+ return 0;
+}

 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_SWAP

--
1.7.1
```