Subject: [PATCH -mm 3/3] i/o accounting and control
Posted by Andrea Righi on Tue, 22 Jul 2008 20:58:59 GMT
View Forum Message <> Reply to Message

Apply the io-throttle controller to the opportune kernel functions. Both
accounting and throttling functionalities are performed by
cgroup_io_throttle().

Signed-off-by: Andrea Righi <righi.andrea@gmail.com>
---
 block/blk-core.c | 9 +++++++++
 fs/aio.c | 31 +++++++++++++++++++++++++++++++-
 fs/buffer.c | 20 ++++++++++++++++++---
 fs/direct-io.c | 4 ++++
 include/linux/sched.h | 3 +++
 kernel/fork.c | 3 +++
 mm/filemap.c | 18 ++++++++++++++++-
 mm/page-writeback.c | 30 +++++++++++++++++++++++++++---
 mm/readahead.c | 5 +++++
 9 files changed, 115 insertions(+), 8 deletions(-)

diff --git a/block/blk-core.c b/block/blk-core.c
index 4c222ba..431294f 100644
--- a/block/blk-core.c
+++ b/block/blk-core.c
@@ -26,6 +26,7 @@
 #include <linux/swap.h>
 #include <linux/writeback.h>
 #include <linux/task_io_accounting_ops.h>
+#include <linux/blk-io-throttle.h>
 #include <linux/interrupt.h>
 #include <linux/cpu.h>
 #include <linux/blktrace_api.h>
@@ -1482,7 +1483,15 @@ void submit_bio(int rw, struct bio *bio)
   if (rw & WRITE) {
    count_vm_events(PGPGOUT, count);
   } else {
+   struct page *page = bio_iovec_idx(bio, 0)->bv_page;
+
    task_io_account_read(bio->bi_size);
+   /*
+   * Do not throttle page requests that need to be
+   * urgently reclaimed.
+   */
+   cgroup_io_throttle(bio->bi_bdev, bio->bi_size,
+   !(PageReclaim(page) || PageSwapCache(page)));
    count_vm_events(PGPGIN, count);
   }

```
diff --git a/fs/aio.c b/fs/aio.c
index 0051fd9..1f3abb3 100644
--- a/fs/aio.c
+++ b/fs/aio.c
@@ -22,6 +22,7 @@
 #include <linux/sched.h>
 #include <linux/fs.h>
 #include <linux/file.h>
+#include <linux/blk-io-throttle.h>
 #include <linux/mm.h>
 #include <linux/mman.h>
 #include <linux/slab.h>
@@ -1558,6 +1559,8 @@ static int io_submit_one(struct kioctx *ctx, struct iocb __user
*user_iocb,
 {
  struct kiocb *req;
  struct file *file;
+ struct block_device *bdev;
+ struct inode *inode;
  ssize_t ret;

  /* enforce forwards compatibility on users */
@@ -1580,10 +1583,26 @@ static int io_submit_one(struct kioctx *ctx, struct iocb __user
*user_iocb,
  if (unlikely(!file))
   return -EBADF;

+ /*
+  * Pre-account AIO activity: we over-account *all* the bytes here;
+  * bytes read from the page cache and bytes written in already dirtied
+  * pages (that do not generate real i/o on block devices) will be
+  * subtracted later, following the path of aio_run_iocb().
+  */
+ inode = file->f_mapping->host;
+ bdev = inode->i_sb->s_bdev;
+ ret = cgroup_io_throttle(bdev, iocb->aio_nbytes, 0);
+ if (unlikely(ret)) {
+  fput(file);
+  ret = -EAGAIN;
+  goto out_cgroup_io_throttle;
+ }
+
  req = aio_get_req(ctx);  /* returns with 2 references to req */
  if (unlikely(!req)) {
   fput(file);
-  return -EAGAIN;
+  ret = -EAGAIN;
```

```
+  goto out_cgroup_io_throttle;
  }
  req->ki_filp = file;
  if (iocb->aio_flags & IOCB_FLAG_RESFD) {
@@ -1622,12 +1641,14 @@ static int io_submit_one(struct kioctx *ctx, struct iocb __user
*user_iocb,
   goto out_put_req;

  spin_lock_irq(&ctx->ctx_lock);
+ set_in_aio();
  aio_run_iocb(req);
  if (!list_empty(&ctx->run_list)) {
   /* drain the run list */
   while (__aio_run_iocbs(ctx))
    ;
  }
+ unset_in_aio();
  spin_unlock_irq(&ctx->ctx_lock);
  aio_put_req(req); /* drop extra ref to req */
  return 0;
@@ -1635,6 +1656,8 @@ static int io_submit_one(struct kioctx *ctx, struct iocb __user
*user_iocb,
 out_put_req:
  aio_put_req(req); /* drop extra ref to req */
  aio_put_req(req); /* drop i/o ref to req */
+out_cgroup_io_throttle:
+ cgroup_io_throttle(bdev, -iocb->aio_nbytes, 0);
  return ret;
 }

@@ -1746,6 +1769,12 @@ asmlinkage long sys_io_cancel(aio_context_t ctx_id, struct iocb
__user *iocb,
  ret = -EAGAIN;
  kiocb = lookup_kiocb(ctx, iocb, key);
  if (kiocb && kiocb->ki_cancel) {
+  struct block_device *bdev;
+  struct inode *inode = kiocb->ki_filp->f_mapping->host;
+
+  bdev = inode->i_sb->s_bdev;
+  cgroup_io_throttle(bdev, -kiocb->ki_nbytes, 0);
+
   cancel = kiocb->ki_cancel;
   kiocb->ki_users ++;
   kiocbSetCancelled(kiocb);
diff --git a/fs/buffer.c b/fs/buffer.c
index 4ffb5bb..6d4bf2c 100644
--- a/fs/buffer.c
+++ b/fs/buffer.c
```

```
@@ -35,6 +35,7 @@
 #include <linux/suspend.h>
 #include <linux/buffer_head.h>
 #include <linux/task_io_accounting_ops.h>
+#include <linux/blk-io-throttle.h>
 #include <linux/bio.h>
 #include <linux/notifier.h>
 #include <linux/cpu.h>
@@ -708,11 +709,14 @@ EXPORT_SYMBOL(mark_buffer_dirty_inode);
 static int __set_page_dirty(struct page *page,
   struct address_space *mapping, int warn)
 {
+ ssize_t cgroup_io_acct = 0;
+ int ret = 0;
+
  if (unlikely(!mapping))
   return !TestSetPageDirty(page);

  if (TestSetPageDirty(page))
-  return 0;
+  goto out;

  spin_lock_irq(&mapping->tree_lock);
  if (page->mapping) { /* Race with truncate? */
@@ -723,14 +727,24 @@ static int __set_page_dirty(struct page *page,
    __inc_bdi_stat(mapping->backing_dev_info,
     BDI_RECLAIMABLE);
    task_io_account_write(PAGE_CACHE_SIZE);
+   cgroup_io_acct = PAGE_CACHE_SIZE;
   }
   radix_tree_tag_set(&mapping->page_tree,
    page_index(page), PAGECACHE_TAG_DIRTY);
  }
  spin_unlock_irq(&mapping->tree_lock);
  __mark_inode_dirty(mapping->host, I_DIRTY_PAGES);
-
- return 1;
+ ret = 1;
+out:
+ if (is_in_aio() && !cgroup_io_acct)
+  cgroup_io_acct = -PAGE_CACHE_SIZE;
+ if (cgroup_io_acct) {
+  struct block_device *bdev = (mapping->host &&
+    mapping->host->i_sb->s_bdev) ?
+    mapping->host->i_sb->s_bdev : NULL;
+  cgroup_io_throttle(bdev, cgroup_io_acct, 0);
+ }
+ return ret;
```

```
 }

 /*
diff --git a/fs/direct-io.c b/fs/direct-io.c
index 9606ee8..f5dcb91 100644
--- a/fs/direct-io.c
+++ b/fs/direct-io.c
@@ -35,6 +35,7 @@
 #include <linux/buffer_head.h>
 #include <linux/rwsem.h>
 #include <linux/uio.h>
+#include <linux/blk-io-throttle.h>
 #include <asm/atomic.h>

 /*
@@ -660,6 +661,9 @@ submit_page_section(struct dio *dio, struct page *page,
   /*
    * Read accounting is performed in submit_bio()
    */
+  struct block_device *bdev = dio->bio ?
+      dio->bio->bi_bdev : NULL;
+  cgroup_io_throttle(bdev, len, 1);
   task_io_account_write(len);
  }

diff --git a/include/linux/sched.h b/include/linux/sched.h
index ba43675..9d4c755 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1250,6 +1250,9 @@ struct task_struct {
  u64 rchar, wchar, syscr, syscw;
 #endif
  struct task_io_accounting ioac;
+#ifdef CONFIG_CGROUP_IO_THROTTLE
+ atomic_t in_aio;
+#endif
 #if defined(CONFIG_TASK_XACCT)
  u64 acct_rss_mem1; /* accumulated rss usage */
  u64 acct_vm_mem1; /* accumulated virtual memory usage */
diff --git a/kernel/fork.c b/kernel/fork.c
index aed1ff7..f8cf5da 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1029,6 +1029,9 @@ static struct task_struct *copy_process(unsigned long clone_flags,
  task_io_accounting_init(p);
  acct_clear_integrals(p);

+#ifdef CONFIG_CGROUP_IO_THROTTLE
```

```
+ atomic_set(&p->in_aio, 0);
+#endif
  p->it_virt_expires = cputime_zero;
  p->it_prof_expires = cputime_zero;
  p->it_sched_expires = 0;
diff --git a/mm/filemap.c b/mm/filemap.c
index 7567d86..bb80789 100644
--- a/mm/filemap.c
+++ b/mm/filemap.c
@@ -13,6 +13,7 @@
 #include <linux/slab.h>
 #include <linux/compiler.h>
 #include <linux/fs.h>
+#include <linux/blk-io-throttle.h>
 #include <linux/uaccess.h>
 #include <linux/aio.h>
 #include <linux/capability.h>
@@ -1011,6 +1012,7 @@ static void do_generic_file_read(struct file *filp, loff_t *ppos,
  pgoff_t prev_index;
  unsigned long offset;      /* offset into pagecache page */
  unsigned int prev_offset;
+ int was_page_ok = 0;
  int error;

  index = *ppos >> PAGE_CACHE_SHIFT;
@@ -1023,7 +1025,8 @@ static void do_generic_file_read(struct file *filp, loff_t *ppos,
   struct page *page;
   pgoff_t end_index;
   loff_t isize;
-  unsigned long nr, ret;
+  ssize_t nr;
+  unsigned long ret;

   cond_resched();
 find_page:
@@ -1051,6 +1054,8 @@ find_page:
         desc, offset))
     goto page_not_up_to_date_locked;
    unlock_page(page);
+  } else {
+   was_page_ok = 1;
   }
 page_ok:
   /*
@@ -1080,6 +1085,17 @@ page_ok:
   }
   nr = nr - offset;
```

```
+  /*
+   * De-account i/o in case of AIO read from the page cache.
+   * AIO accounting was performed in io_submit_one().
+   */
+  if (is_in_aio() && was_page_ok) {
+   struct block_device *bdev = (inode &&
+       inode->i_sb->s_bdev) ?
+       inode->i_sb->s_bdev : NULL;
+   cgroup_io_throttle(bdev, -nr, 0);
+  }
+
    /* If users can be writing to this page using arbitrary
     * virtual addresses, take care about potential aliasing
     * before reading the page on the kernel side.
diff --git a/mm/page-writeback.c b/mm/page-writeback.c
index 29b1d1e..c6207de 100644
--- a/mm/page-writeback.c
+++ b/mm/page-writeback.c
@@ -23,6 +23,7 @@
 #include <linux/init.h>
 #include <linux/backing-dev.h>
 #include <linux/task_io_accounting_ops.h>
+#include <linux/blk-io-throttle.h>
 #include <linux/blkdev.h>
 #include <linux/mpage.h>
 #include <linux/rmap.h>
@@ -430,6 +431,9 @@ static void balance_dirty_pages(struct address_space *mapping)
  unsigned long write_chunk = sync_writeback_pages();

  struct backing_dev_info *bdi = mapping->backing_dev_info;
+ struct block_device *bdev = (mapping->host &&
+     mapping->host->i_sb->s_bdev) ?
+     mapping->host->i_sb->s_bdev : NULL;

  for (;;) {
   struct writeback_control wbc = {
@@ -512,6 +516,14 @@ static void balance_dirty_pages(struct address_space *mapping)
   return;  /* pdflush is already working this queue */

  /*
+  * Apply the cgroup i/o throttling limitations. The accounting of write
+  * activity in page cache is performed in __set_page_dirty(), but since
+  * we cannot sleep there, 0 bytes are accounted here and the function
+  * is invoked only for throttling purpose.
+  */
+ cgroup_io_throttle(bdev, 0, 1);
+
+ /*
```

```
   * In laptop mode, we wait until hitting the higher threshold before
   * starting background writeout, and then write out all the way down
   * to the lower threshold.  So slow writers cause minimal disk activity.
@@ -1074,8 +1086,11 @@ int __set_page_dirty_no_writeback(struct page *page)
  */
 int __set_page_dirty_nobuffers(struct page *page)
 {
+ struct address_space *mapping = page_mapping(page);
+ ssize_t cgroup_io_acct = 0;
+ int ret = 0;
+
   if (!TestSetPageDirty(page)) {
-  struct address_space *mapping = page_mapping(page);
    struct address_space *mapping2;

    if (!mapping)
@@ -1091,6 +1106,7 @@ int __set_page_dirty_nobuffers(struct page *page)
      __inc_bdi_stat(mapping->backing_dev_info,
        BDI_RECLAIMABLE);
      task_io_account_write(PAGE_CACHE_SIZE);
+     cgroup_io_acct = PAGE_CACHE_SIZE;
    }
    radix_tree_tag_set(&mapping->page_tree,
     page_index(page), PAGECACHE_TAG_DIRTY);
@@ -1100,9 +1116,17 @@ int __set_page_dirty_nobuffers(struct page *page)
    /* !PageAnon && !swapper_space */
    __mark_inode_dirty(mapping->host, I_DIRTY_PAGES);
   }
-  return 1;
+  ret = 1;
  }
- return 0;
+ if (is_in_aio() && !cgroup_io_acct)
+  cgroup_io_acct = -PAGE_CACHE_SIZE;
+ if (cgroup_io_acct) {
+  struct block_device *bdev = (mapping->host &&
+      mapping->host->i_sb->s_bdev) ?
+      mapping->host->i_sb->s_bdev : NULL;
+  cgroup_io_throttle(bdev, cgroup_io_acct, 0);
+ }
+ return ret;
 }
 EXPORT_SYMBOL(__set_page_dirty_nobuffers);

diff --git a/mm/readahead.c b/mm/readahead.c
index 137bc56..448f065 100644
--- a/mm/readahead.c
+++ b/mm/readahead.c
```

```
@@ -14,6 +14,7 @@
 #include <linux/blkdev.h>
 #include <linux/backing-dev.h>
 #include <linux/task_io_accounting_ops.h>
+#include <linux/blk-io-throttle.h>
 #include <linux/pagevec.h>
 #include <linux/pagemap.h>

@@ -58,6 +59,9 @@ int read_cache_pages(struct address_space *mapping, struct list_head
*pages,
    int (*filler)(void *, struct page *), void *data)
 {
  struct page *page;
+ struct block_device *bdev =
+  (mapping->host && mapping->host->i_sb->s_bdev) ?
+  mapping->host->i_sb->s_bdev : NULL;
  int ret = 0;

  while (!list_empty(pages)) {
@@ -76,6 +80,7 @@ int read_cache_pages(struct address_space *mapping, struct list_head
*pages,
    break;
   }
   task_io_account_read(PAGE_CACHE_SIZE);
+  cgroup_io_throttle(bdev, PAGE_CACHE_SIZE, 1);
  }
  return ret;
 }
--
1.5.4.3
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers