
Subject: [PATCH 08/11] fuse: use req->page_descs[] for argpages cases
Posted by [Maxim Patlasov](#) on Wed, 19 Sep 2012 16:33:03 GMT

[View Forum Message](#) <> [Reply to Message](#)

Previously, anyone who set flag 'argpages' only filled req->pages[] and set per-request page_offset. This patch re-works all cases where argpages=1 to fill req->page_descs[] properly.

Having req->page_descs[] filled properly allows to re-work fuse_copy_pages() to copy page fragments described by req->page_descs[]. This will be useful for next patches optimizing direct_IO.

Signed-off-by: Maxim Patlasov <mpatlasov@parallels.com>

```
fs/fuse/cuse.c | 1 +
fs/fuse/dev.c | 7 +++----
fs/fuse/dir.c | 1 +
fs/fuse/file.c | 20 ++++++
4 files changed, 25 insertions(+), 4 deletions(-)
```

```
diff --git a/fs/fuse/cuse.c b/fs/fuse/cuse.c
```

```
index 2f1d5dd..38efe69 100644
```

```
--- a/fs/fuse/cuse.c
```

```
+++ b/fs/fuse/cuse.c
```

```
@@ -441,6 +441,7 @@ static int cuse_send_init(struct cuse_conn *cc)
```

```
    req->out.argvar = 1;
```

```
    req->out.argpages = 1;
```

```
    req->pages[0] = page;
```

```
+ req->page_descs[0].length = req->out.args[1].size;
```

```
    req->num_pages = 1;
```

```
    req->end = cuse_process_init_reply;
```

```
    fuse_request_send_background(fc, req);
```

```
diff --git a/fs/fuse/dev.c b/fs/fuse/dev.c
```

```
index ea63d39..21b6272 100644
```

```
--- a/fs/fuse/dev.c
```

```
+++ b/fs/fuse/dev.c
```

```
@@ -888,11 +888,11 @@ static int fuse_copy_pages(struct fuse_copy_state *cs, unsigned
nbytes,
```

```
{
```

```
    unsigned i;
```

```
    struct fuse_req *req = cs->req;
```

```
- unsigned offset = req->page_descs[0].offset;
```

```
- unsigned count = min(nbytes, (unsigned) PAGE_SIZE - offset);
```

```
    for (i = 0; i < req->num_pages && (nbytes || zeroing); i++) {
```

```
        int err;
```

```
+ unsigned offset = req->page_descs[i].offset;
```

```
+ unsigned count = min(nbytes, req->page_descs[i].length);
```

```

    err = fuse_copy_page(cs, &req->pages[i], offset, count,
        zeroing);
@@ -900,8 +900,6 @@ static int fuse_copy_pages(struct fuse_copy_state *cs, unsigned nbytes,
    return err;

    nbytes -= count;
-   count = min(nbytes, (unsigned) PAGE_SIZE);
-   offset = 0;
}
return 0;
}
@@ -1611,6 +1609,7 @@ static int fuse_retrieve(struct fuse_conn *fc, struct inode *inode,

    this_num = min_t(unsigned, num, PAGE_CACHE_SIZE - offset);
    req->pages[req->num_pages] = page;
+   req->page_descs[req->num_pages].length = this_num;
    req->num_pages++;

    num -= this_num;
diff --git a/fs/fuse/dir.c b/fs/fuse/dir.c
index 1929bfb..95e4016 100644
--- a/fs/fuse/dir.c
+++ b/fs/fuse/dir.c
@@ -1179,6 +1179,7 @@ static int fuse_readdir(struct file *file, void *dstbuf, filldir_t filldir)
    req->out.argpages = 1;
    req->num_pages = 1;
    req->pages[0] = page;
+   req->page_descs[0].length = PAGE_SIZE;
    fuse_read_fill(req, file, file->f_pos, PAGE_SIZE, FUSE_READDIR);
    fuse_request_send(fc, req);
    nbytes = req->out.args[0].size;
diff --git a/fs/fuse/file.c b/fs/fuse/file.c
index b68cf3b..8a1f833 100644
--- a/fs/fuse/file.c
+++ b/fs/fuse/file.c
@@ -555,6 +555,7 @@ static int fuse_readpage(struct file *file, struct page *page)
    req->out.argpages = 1;
    req->num_pages = 1;
    req->pages[0] = page;
+   req->page_descs[0].length = count;
    num_read = fuse_send_read(req, file, pos, count, NULL);
    err = req->out.h.error;
    fuse_put_request(fc, req);
@@ -674,6 +675,7 @@ static int fuse_readpages_fill(void *_data, struct page *page)

    page_cache_get(page);
    req->pages[req->num_pages] = page;

```

```

+ req->page_descs[req->num_pages].length = PAGE_SIZE;
  req->num_pages++;
  data->nr_pages--;
  return 0;
@@ -869,6 +871,7 @@ static ssize_t fuse_fill_write_pages(struct fuse_req *req,

  err = 0;
  req->pages[req->num_pages] = page;
+ req->page_descs[req->num_pages].length = tmp;
  req->num_pages++;

  iov_iter_advance(ii, tmp);
@@ -1044,6 +1047,15 @@ static void fuse_release_user_pages(struct fuse_req *req, int write)
}
}

+static inline void fuse_page_descs_length_init(struct fuse_req *req)
+{
+ int i;
+
+ for (i = 0; i < req->num_pages; i++)
+ req->page_descs[i].length = PAGE_SIZE -
+ req->page_descs[i].offset;
+}
+
static int fuse_get_user_pages(struct fuse_req *req, const char __user *buf,
                             size_t *nbytesp, int write)
{
@@ -1071,6 +1083,7 @@ static int fuse_get_user_pages(struct fuse_req *req, const char __user
*buf,

  req->num_pages = npages;
  req->page_descs[0].offset = offset;
+ fuse_page_descs_length_init(req);

  if (write)
    req->in.argpages = 1;
@@ -1078,6 +1091,11 @@ static int fuse_get_user_pages(struct fuse_req *req, const char
__user *buf,
  req->out.argpages = 1;

  nbytes = (req->num_pages << PAGE_SHIFT) - req->page_descs[0].offset;
+
+ if (*nbytesp < nbytes)
+ req->page_descs[req->num_pages - 1].length -=
+ nbytes - *nbytesp;
+
  *nbytesp = min(*nbytesp, nbytes);

```

```
return 0;
@@ -1315,6 +1333,7 @@ static int fuse_writepage_locked(struct page *page)
    req->num_pages = 1;
    req->pages[0] = tmp_page;
    req->page_descs[0].offset = 0;
+ req->page_descs[0].length = PAGE_SIZE;
    req->end = fuse_writepage_end;
    req->inode = inode;

@@ -1901,6 +1920,7 @@ long fuse_do_ioctl(struct file *file, unsigned int cmd, unsigned long
arg,
}
memcpy(req->pages, pages, sizeof(req->pages[0]) * num_pages);
req->num_pages = num_pages;
+ fuse_page_descs_length_init(req);

/* okay, let's send it to the client */
req->in.h.opcode = FUSE_IOCTL;
```
