

---

Subject: [PATCH 10/12] fuse: pass iov[] to fuse\_get\_user\_pages()  
Posted by [Maxim Patlasov](#) on Fri, 26 Oct 2012 15:50:15 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

The patch makes preliminary work for the next patch optimizing scatter-gather direct IO. The idea is to allow fuse\_get\_user\_pages() to pack as many iov-s to each fuse request as possible. So, here we only rework all related call-paths to carry iov[] from fuse\_direct\_IO() to fuse\_get\_user\_pages().

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

---

fs/fuse/file.c | 103 ++++++-----  
1 files changed, 49 insertions(+), 54 deletions(-)

diff --git a/fs/fuse/file.c b/fs/fuse/file.c

index 51996af..9934321 100644

--- a/fs/fuse/file.c

+++ b/fs/fuse/file.c

```
@@ -1056,14 +1056,18 @@ static inline void fuse_page_descs_length_init(struct fuse_req *req)
    req->page_descs[i].offset;
}
```

```
-static int fuse_get_user_pages(struct fuse_req *req, const char __user *buf,
```

```
+static int fuse_get_user_pages(struct fuse_req *req, struct iov_iter *ii,
    size_t *nbytesp, int write)
```

```
{
    size_t nbytes = *nbytesp;
- unsigned long user_addr = (unsigned long) buf;
- unsigned offset = user_addr & ~PAGE_MASK;
+ size_t frag_size = min(iov_iter_single_seg_count(ii), nbytes);
+ unsigned long user_addr;
+ unsigned offset;
    int npages;
```

```
+ user_addr = (unsigned long)ii->iov->iov_base + ii->iov_offset;
```

```
+ offset = user_addr & ~PAGE_MASK;
```

```
+
```

```
/* Special case for kernel I/O: can copy directly into the buffer */
```

```
if (segment_eq(get_fs(), KERNEL_DS)) {
    if (write)
```

```
@@ -1071,10 +1075,12 @@ static int fuse_get_user_pages(struct fuse_req *req, const char
    __user *buf,
```

```
else
```

```
    req->out.args[0].value = (void *) user_addr;
```

```
+ iov_iter_advance(ii, frag_size);
```

```
+ *nbytesp = frag_size;
```

```
return 0;
```

```

}

- nbytes = min_t(size_t, nbytes, FUSE_MAX_PAGES_PER_REQ << PAGE_SHIFT);
+ nbytes = min_t(size_t, frag_size, FUSE_MAX_PAGES_PER_REQ << PAGE_SHIFT);
  npages = (nbytes + offset + PAGE_SIZE - 1) >> PAGE_SHIFT;
  npages = clamp(npages, 1, FUSE_MAX_PAGES_PER_REQ);
  npages = get_user_pages_fast(user_addr, npages, !write, req->pages);
@@ -1092,17 +1098,19 @@ static int fuse_get_user_pages(struct fuse_req *req, const char
__user *buf,

    nbytes = (req->num_pages << PAGE_SHIFT) - req->page_descs[0].offset;

- if (*nbytesp < nbytes)
+ if (frag_size < nbytes)
    req->page_descs[req->num_pages - 1].length -=
-   nbytes - *nbytesp;
+   nbytes - frag_size;

- *nbytesp = min(*nbytesp, nbytes);
+ *nbytesp = min(frag_size, nbytes);
+ iov_iter_advance(ii, *nbytesp);

    return 0;
}

-ssize_t fuse_direct_io(struct file *file, const char __user *buf,
-   size_t count, loff_t *ppos, int write)
+static ssize_t __fuse_direct_io(struct file *file, const struct iovec *iov,
+   unsigned long nr_segs, size_t count,
+   loff_t *ppos, int write)
{
    struct fuse_file *ff = file->private_data;
    struct fuse_conn *fc = ff->fc;
@@ -1110,6 +1118,9 @@ ssize_t fuse_direct_io(struct file *file, const char __user *buf,
    loff_t pos = *ppos;
    ssize_t res = 0;
    struct fuse_req *req;
+ struct iov_iter ii;
+
+ iov_iter_init(&ii, iov, nr_segs, count, 0);

    req = fuse_get_req(fc, FUSE_MAX_PAGES_PER_REQ);
    if (IS_ERR(req))
@@ -1119,7 +1130,7 @@ ssize_t fuse_direct_io(struct file *file, const char __user *buf,
    size_t nres;
    fl_owner_t owner = current->files;
    size_t nbytes = min(count, nmax);
- int err = fuse_get_user_pages(req, buf, &nbytes, write);

```

```

+ int err = fuse_get_user_pages(req, &ii, &nbytes, write);
  if (err) {
    res = err;
    break;
@@ -1142,7 +1153,6 @@ ssize_t fuse_direct_io(struct file *file, const char __user *buf,
  count -= nres;
  res += nres;
  pos += nres;
- buf += nres;
  if (nres != nbytes)
    break;
  if (count) {
@@ -1159,10 +1169,17 @@ ssize_t fuse_direct_io(struct file *file, const char __user *buf,

  return res;
}
+
+ssize_t fuse_direct_io(struct file *file, const char __user *buf,
+  size_t count, loff_t *ppos, int write)
+{
+ struct iovec iov = { .iov_base = (void *)buf, .iov_len = count };
+ return __fuse_direct_io(file, &iov, 1, count, ppos, write);
+}
EXPORT_SYMBOL_GPL(fuse_direct_io);

-static ssize_t fuse_direct_read(struct file *file, char __user *buf,
-  size_t count, loff_t *ppos)
+static ssize_t __fuse_direct_read(struct file *file, const struct iovec *iov,
+  unsigned long nr_segs, loff_t *ppos)
{
  ssize_t res;
  struct inode *inode = file->f_path.dentry->d_inode;
@@ -1170,22 +1187,31 @@ static ssize_t fuse_direct_read(struct file *file, char __user *buf,
  if (is_bad_inode(inode))
    return -EIO;

- res = fuse_direct_io(file, buf, count, ppos, 0);
+ res = __fuse_direct_io(file, iov, nr_segs, iov_length(iov, nr_segs),
+  ppos, 0);

  fuse_invalidate_attr(inode);

  return res;
}

-static ssize_t __fuse_direct_write(struct file *file, const char __user *buf,
-  size_t count, loff_t *ppos)
+static ssize_t fuse_direct_read(struct file *file, char __user *buf,

```

```

+     size_t count, loff_t *ppos)
+{
+ struct iovec iov = { .iov_base = (void *)buf, .iov_len = count };
+ return __fuse_direct_read(file, &iov, 1, ppos);
+}
+
+static ssize_t __fuse_direct_write(struct file *file, const struct iovec *iov,
+     unsigned long nr_segs, loff_t *ppos)
+{
+     struct inode *inode = file->f_path.dentry->d_inode;
+ size_t count = iov_length(iov, nr_segs);
+     ssize_t res;

+     res = generic_write_checks(file, ppos, &count, 0);
+     if (!res) {
- res = fuse_direct_io(file, buf, count, ppos, 1);
+ res = __fuse_direct_io(file, iov, nr_segs, count, ppos, 1);
+     if (res > 0)
+         fuse_write_update_size(inode, *ppos);
+     }
@@ -1198,6 +1224,7 @@ static ssize_t __fuse_direct_write(struct file *file, const char __user
*buf,
static ssize_t fuse_direct_write(struct file *file, const char __user *buf,
    size_t count, loff_t *ppos)
{
+ struct iovec iov = { .iov_base = (void *)buf, .iov_len = count };
+     struct inode *inode = file->f_path.dentry->d_inode;
+     ssize_t res;

@@ -1206,7 +1233,7 @@ static ssize_t fuse_direct_write(struct file *file, const char __user *buf,

/* Don't allow parallel writes to the same file */
mutex_lock(&inode->i_mutex);
- res = __fuse_direct_write(file, buf, count, ppos);
+ res = __fuse_direct_write(file, &iov, 1, ppos);
+     mutex_unlock(&inode->i_mutex);

+     return res;
@@ -2167,41 +2194,6 @@ int fuse_notify_poll_wakeup(struct fuse_conn *fc,
+     return 0;
+ }

-static ssize_t fuse_loop_dio(struct file *filp, const struct iovec *iov,
-     unsigned long nr_segs, loff_t *ppos, int rw)
-{
- const struct iovec *vector = iov;
-     ssize_t ret = 0;
-

```

```

- while (nr_segs > 0) {
- void __user *base;
- size_t len;
- ssize_t nr;
-
- base = vector->iov_base;
- len = vector->iov_len;
- vector++;
- nr_segs--;
-
- if (rw == WRITE)
- nr = __fuse_direct_write(filp, base, len, ppos);
- else
- nr = fuse_direct_read(filp, base, len, ppos);
-
- if (nr < 0) {
- if (!ret)
- ret = nr;
- break;
- }
- ret += nr;
- if (nr != len)
- break;
- }
-
- return ret;
-}
-
static ssize_t
fuse_direct_IO(int rw, struct kiocb *iocb, const struct iovec *iov,
    loff_t offset, unsigned long nr_segs)
@@ -2213,7 +2205,10 @@ fuse_direct_IO(int rw, struct kiocb *iocb, const struct iovec *iov,
    file = iocb->ki_filp;
    pos = offset;

- ret = fuse_loop_dio(file, iov, nr_segs, &pos, rw);
+ if (rw == WRITE)
+ ret = __fuse_direct_write(file, iov, nr_segs, &pos);
+ else
+ ret = __fuse_direct_read(file, iov, nr_segs, &pos);

    return ret;
}

```

---