
Subject: [PATCH 09/11] fuse: pass iov[] to fuse_get_user_pages()
Posted by [Maxim Patlasov](#) on Wed, 19 Sep 2012 16:33:12 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 | 108 ++++++-----
1 files changed, 55 insertions(+), 53 deletions(-)

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

index 8a1f833..db9efb5 100644

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

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

```
@@ -1056,11 +1056,15 @@ 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,
```

```
+    const struct iovec **iov_pp,
```

```
+    unsigned long *nr_segs_p,
```

```
+    size_t *iov_offset_p,
```

```
+    size_t *nbytesp, int write)
```

```
{
    size_t nbytes = *nbytesp;
```

```
- unsigned long user_addr = (unsigned long) buf;
```

```
+ size_t frag_size = min_t(size_t, nbytes, (*iov_pp)->iov_len - *iov_offset_p);
```

```
+ unsigned long user_addr = (unsigned long)(*iov_pp)->iov_base + *iov_offset_p;
```

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

```
+ int npages;
```

```
@@ -1071,10 +1075,13 @@ static int fuse_get_user_pages(struct fuse_req *req, const char
__user *buf,
```

```
    else
```

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

```
+ (*iov_pp)++;
```

```
+ (*nr_segs_p)--;
```

```
+ *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 +1099,26 @@ 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);
+
+ if (*nbytesp < (*iov_pp)->iov_len - *iov_offset_p) {
+   *iov_offset_p += *nbytesp;
+ } else {
+   (*iov_pp)++;
+   (*nr_segs_p)--;
+   *iov_offset_p = 0;
+ }

    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 +1126,7 @@ ssize_t fuse_direct_io(struct file *file, const char __user *buf,
    loff_t pos = *ppos;
    ssize_t res = 0;
    struct fuse_req *req;
+ size_t iov_offset = 0;

    req = fuse_get_req(fc, FUSE_MAX_PAGES_PER_REQ);
    if (IS_ERR(req))
@@ -1119,7 +1136,8 @@ 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, &iiov, &nr_segs, &iiov_offset,
+       &nbytes, write);
    if (err) {
        res = err;
        break;
@@ -1142,7 +1160,6 @@ 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 +1176,17 @@ 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 iiov = { .iiov_base = (void *)buf, .iiov_len = count };
+   return __fuse_direct_io(file, &iiov, 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 *iiov,
+    unsigned long nr_segs, loff_t *ppos)
{
    ssize_t res;
    struct inode *inode = file->f_path.dentry->d_inode;
@@ -1170,22 +1194,31 @@ 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, iiov, nr_segs, iiov_length(iiov, 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 +1231,7 @@ 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 +1240,7 @@ 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;
@@ -2166,41 +2200,6 @@ 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)
@@ -2212,7 +2211,10 @@ 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;
}

```
