
Subject: [PATCH 1/4] fuse: add basic support of iovec[] to fuse_req
Posted by [Maxim Patlasov](#) on Fri, 20 Jul 2012 11:50:23 GMT
[View Forum Message](#) <> [Reply to Message](#)

The patch allows fuse_req to refer to array of iovec-s describing layout of user-data over req->pages. fuse_copy_pages() is re-worked to support both cased: former layout where pages[] corresponded to <buf, len> and newer one where pages[] corresponds to iovec[].

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

fs/fuse/dev.c | 52 +++
fs/fuse/fuse_i.h | 12 ++++++
2 files changed, 59 insertions(+), 5 deletions(-)

diff --git a/fs/fuse/dev.c b/fs/fuse/dev.c
index 7df2b5e..cdae525 100644

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

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

@@ -850,9 +850,9 @@ static int fuse_copy_page(struct fuse_copy_state *cs, struct page
**pagep,
return 0;
}

/* Copy pages in the request to/from userspace buffer */

-static int fuse_copy_pages(struct fuse_copy_state *cs, unsigned nbytes,
- int zeroing)

+/* Start from addr(pages[0]) + page_offset. No holes in the middle. */

+static int fuse_copy_pages_for_buf(struct fuse_copy_state *cs, unsigned nbytes,
+ int zeroing)

{
unsigned i;
struct fuse_req *req = cs->req;
@@ -874,6 +874,52 @@ static int fuse_copy_pages(struct fuse_copy_state *cs, unsigned
nbytes,
return 0;
}

+/* Take iov_offset as offset in iovec[0]. Iterate based on iovec[i].iov_len */

+static int fuse_copy_pages_for_iovec(struct fuse_copy_state *cs,
+ unsigned nbytes, int zeroing)

+{
+ unsigned i;
+ struct fuse_req *req = cs->req;
+ const struct iovec *iov = req->iovec;
+ unsigned iov_offset = req->iov_offset;
+
+ for (i = 0; i < req->num_pages && (nbytes || zeroing); i++) {

```

+ int err;
+ unsigned long user_addr = (unsigned long)iov->iov_base +
+     iov_offset;
+ unsigned offset = user_addr & ~PAGE_MASK;
+ unsigned count = min_t(size_t, PAGE_SIZE - offset,
+     iov->iov_len - iov_offset);
+ count = min(nbytes, count);
+
+ err = fuse_copy_page(cs, &req->pages[i], offset, count,
+     zeroing);
+ if (err)
+     return err;
+
+ nbytes -= count;
+
+ if (count < iov->iov_len - iov_offset) {
+     iov_offset += count;
+ } else {
+     iov++;
+     iov_offset = 0;
+ }
+ }
+
+ return 0;
+}
+
+/* Copy pages in the request to/from userspace buffer */
+static int fuse_copy_pages(struct fuse_copy_state *cs, unsigned nbytes,
+    int zeroing)
+{
+    if (cs->req->iovec)
+        return fuse_copy_pages_for_iovec(cs, nbytes, zeroing);
+    else
+        return fuse_copy_pages_for_buf(cs, nbytes, zeroing);
+}
+
+/* Copy a single argument in the request to/from userspace buffer */
+static int fuse_copy_one(struct fuse_copy_state *cs, void *val, unsigned size)
+{
diff --git a/fs/fuse/fuse_i.h b/fs/fuse/fuse_i.h
index 771fb63..255b7cd 100644
--- a/fs/fuse/fuse_i.h
+++ b/fs/fuse/fuse_i.h
@@ -296,8 +296,16 @@ struct fuse_req {
    /** number of pages in vector */
    unsigned num_pages;

- /** offset of data on first page */

```

```
- unsigned page_offset;
+ /** If set, it describes layout of user-data in pages[] */
+ const struct iovec *iovec;
+
+ union {
+ /** offset of data on first page */
+ unsigned page_offset;
+
+ /** or in first iovec */
+ unsigned iov_offset;
+ };

/** File used in the request (or NULL) */
struct fuse_file *ff;
```
