Subject: Re: [PATCH 07/11] fuse: add per-page descriptor <offset, length> to fuse_req (v2)
Posted by Miklos Szeredi on Thu, 25 Oct 2012 13:24:17 GMT
View Forum Message <> Reply to Message

Maxim Patlasov <mpatlasov@parallels.com> writes:

> The ability to save page pointers along with lengths and offsets in fuse_req
> will be useful to cover several iovec-s with a single fuse_req.
>
> Per-request page_offset is removed because anybody who need it can use
> req->page_descs[0].offset instead.
>
> Changed in v2:
> - replaced structure page_desc with fuse_page_desc
>
> Signed-off-by: Maxim Patlasov <mpatlasov@parallels.com>
> ---
> fs/fuse/dev.c | 26 ++++++++++++++++++++-------
> fs/fuse/file.c | 10 +++++-----
> fs/fuse/fuse_i.h | 15 ++++++++++---
> 3 files changed, 36 insertions(+), 15 deletions(-)
>
> diff --git a/fs/fuse/dev.c b/fs/fuse/dev.c
> index b241a7d..72ad962 100644
> --- a/fs/fuse/dev.c
> +++ b/fs/fuse/dev.c
> @@ -35,14 +35,17 @@ static struct fuse_conn *fuse_get_conn(struct file *file)
> }
>
> static void fuse_request_init(struct fuse_req *req, struct page **pages,
> +   struct fuse_page_desc *page_descs,
>   unsigned npages)
> {
>   memset(req, 0, sizeof(*req));
> + memset(page_descs, 0, sizeof(*page_descs) * npages);

Makes me wonder: why aren't we zeroing out the page array too?

>   INIT_LIST_HEAD(&req->list);
>   INIT_LIST_HEAD(&req->intr_entry);
>   init_waitqueue_head(&req->waitq);
>   atomic_set(&req->count, 1);
>   req->pages = pages;
> + req->page_descs = page_descs;
>   req->max_pages = npages;
> }
>

```
> @@ -51,18 +54,25 @@ static struct fuse_req *__fuse_request_alloc(int npages, gfp_t flags)
>   struct fuse_req *req = kmem_cache_alloc(fuse_req_cachep, flags);
>   if (req) {
>    struct page **pages;
> +  struct fuse_page_desc *page_descs;
>
> -  if (npages <= FUSE_REQ_INLINE_PAGES)
> +  if (npages <= FUSE_REQ_INLINE_PAGES) {
>     pages = req->inline_pages;
> -  else
> +   page_descs = req->inline_page_descs;
> +  } else {
>     pages = kmalloc(sizeof(struct page *) * npages, flags);
> +   page_descs = kmalloc(sizeof(struct fuse_page_desc) *
> +          npages, flags);
> +  }
>
> -  if (!pages) {
> +  if (!pages || !page_descs) {
> +   kfree(pages);
> +   kfree(page_descs);
>     kmem_cache_free(fuse_req_cachep, req);
>     return NULL;
>    }
>
> -  fuse_request_init(req, pages, npages);
> +  fuse_request_init(req, pages, page_descs, npages);
>   }
>   return req;
>  }
> @@ -82,6 +92,8 @@ void fuse_request_free(struct fuse_req *req)
>  {
>   if (req->pages != req->inline_pages)
>    kfree(req->pages);
> + if (req->page_descs != req->inline_page_descs)
> +  kfree(req->page_descs);
```

You allocate them together, you can free them together. Just add a
BUG_ON() if you feel paranoid.

```
>   kmem_cache_free(fuse_req_cachep, req);
>  }
>
> @@ -186,7 +198,7 @@ static void put_reserved_req(struct fuse_conn *fc, struct fuse_req *req)
>   struct fuse_file *ff = file->private_data;
>
>   spin_lock(&fc->lock);
> - fuse_request_init(req, req->pages, req->max_pages);
```

> + fuse_request_init(req, req->pages, req->page_descs, req->max_pages);
>   BUG_ON(ff->reserved_req);
>   ff->reserved_req = req;
>   wake_up_all(&fc->reserved_req_waitq);
> @@ -877,7 +889,7 @@ static int fuse_copy_pages(struct fuse_copy_state *cs, unsigned
nbytes,
> {
>   unsigned i;
>   struct fuse_req *req = cs->req;
> - unsigned offset = req->page_offset;
> + 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++) {
> @@ -1585,7 +1597,7 @@ static int fuse_retrieve(struct fuse_conn *fc, struct inode *inode,
>   req->in.h.nodeid = outarg->nodeid;
>   req->in.numargs = 2;
>   req->in.argpages = 1;
> - req->page_offset = offset;
> + req->page_descs[0].offset = offset;
>   req->end = fuse_retrieve_end;
>
>   index = outarg->offset >> PAGE_CACHE_SHIFT;
> diff --git a/fs/fuse/file.c b/fs/fuse/file.c
> index 08899a6..b68cf3b 100644
> --- a/fs/fuse/file.c
> +++ b/fs/fuse/file.c
> @@ -798,7 +798,7 @@ static size_t fuse_send_write_pages(struct fuse_req *req, struct file
*file,
>
>   res = fuse_send_write(req, file, pos, count, NULL);
>
> - offset = req->page_offset;
> + offset = req->page_descs[0].offset;
>   count = res;
>   for (i = 0; i < req->num_pages; i++) {
>   struct page *page = req->pages[i];
> @@ -829,7 +829,7 @@ static ssize_t fuse_fill_write_pages(struct fuse_req *req,
>   int err;
>
>   req->in.argpages = 1;
> - req->page_offset = offset;
> + req->page_descs[0].offset = offset;
>
>   do {
>   size_t tmp;
> @@ -1070,14 +1070,14 @@ static int fuse_get_user_pages(struct fuse_req *req, const char
__user *buf,

```
>    return npages;
>
>   req->num_pages = npages;
> - req->page_offset = offset;
> + req->page_descs[0].offset = offset;
>
>   if (write)
>    req->in.argpages = 1;
>   else
>    req->out.argpages = 1;
>
> - nbytes = (req->num_pages << PAGE_SHIFT) - req->page_offset;
> + nbytes = (req->num_pages << PAGE_SHIFT) - req->page_descs[0].offset;
>   *nbytesp = min(*nbytesp, nbytes);
>
>   return 0;
> @@ -1314,7 +1314,7 @@ static int fuse_writepage_locked(struct page *page)
>   req->in.argpages = 1;
>   req->num_pages = 1;
>   req->pages[0] = tmp_page;
> - req->page_offset = 0;
> + req->page_descs[0].offset = 0;
>   req->end = fuse_writepage_end;
>   req->inode = inode;
>
> diff --git a/fs/fuse/fuse_i.h b/fs/fuse/fuse_i.h
> index fd69d62..c07e454 100644
> --- a/fs/fuse/fuse_i.h
> +++ b/fs/fuse/fuse_i.h
> @@ -203,6 +203,12 @@ struct fuse_out {
>   struct fuse_arg args[3];
>  };
>
> +/** FUSE page descriptor */
> +struct fuse_page_desc {
> + unsigned int length;
> + unsigned int offset;
> +};
> +
>  /** The request state */
>  enum fuse_req_state {
>   FUSE_REQ_INIT = 0,
> @@ -296,18 +302,21 @@ struct fuse_req {
>   /** page vector */
>   struct page **pages;
>
> + /** page-descriptor vector */
> + struct fuse_page_desc *page_descs;
```

```
> +
>   /** size of the 'pages' array */
>   unsigned max_pages;
>
>   /** inline page vector */
>   struct page *inline_pages[FUSE_REQ_INLINE_PAGES];
>
> + /** inline page-descriptor vector */
> + struct fuse_page_desc inline_page_descs[FUSE_REQ_INLINE_PAGES];
> +
>   /** number of pages in vector */
>   unsigned num_pages;
>
> - /** offset of data on first page */
> - unsigned page_offset;
> -
>   /** File used in the request (or NULL) */
>   struct fuse_file *ff;
>
```