
Subject: [PATH][RFC] mm: Move common segments checks to separate function
Posted by [Dmitriy Monakhov](#) on Wed, 24 Jan 2007 18:57:21 GMT

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

Move common segments checks from `__generic_file_aio_{read,write}_nolock()`
to separate helper function `generic_segment_checks()`.

Signed-off-by: Dmitriy Monakhov <dmonakhov@openvz.org>

```
-----  
  
diff --git a/mm/filemap.c b/mm/filemap.c  
index 8332c77..d01abb6 100644  
--- a/mm/filemap.c  
+++ b/mm/filemap.c  
@@ -1126,6 +1126,38 @@ success:  
     return size;  
 }  
  
+/*  
+ * Performs necessary segments checks.  
+ * Returns bytes count or appropriate error code that caller should return.  
+ */  
+static ssize_t  
+generic_segment_checks(unsigned long nr_segs, const struct iovec *iov,  
+ unsigned long access_flag)  
+{  
+ size_t bcount;  
+ unsigned long seg;  
+ bcount = 0;  
+ for (seg = 0; seg < nr_segs; seg++) {  
+ const struct iovec *iv = &iov[seg];  
+  
+ /*  
+ * If any segment has a negative length, or the cumulative  
+ * length ever wraps negative then return -EINVAL.  
+ */  
+ bcount += iv->iov_len;  
+ if (unlikely((ssize_t)(bcount|iv->iov_len) < 0))  
+ return -EINVAL;  
+ if (access_ok(access_flag, iv->iov_base, iv->iov_len))  
+ continue;  
+ if (seg == 0)  
+ return -EFAULT;  
+ nr_segs = seg;  
+ bcount -= iv->iov_len; /* This segment is no good */  
+ break;  
+ }  
+ return bcount;
```

```

+}
+
/**
 * generic_file_aio_read - generic filesystem read routine
 * @iocb: kernel I/O control block
@@ -1146,25 +1178,7 @@ generic_file_aio_read(struct kiocb *iocb
 size_t count;
 loff_t *ppos = &iocb->ki_pos;

- count = 0;
- for (seg = 0; seg < nr_segs; seg++) {
- const struct iovec *iv = &iov[seg];
-
- /*
-  * If any segment has a negative length, or the cumulative
-  * length ever wraps negative then return -EINVAL.
-  */
- count += iv->iov_len;
- if (unlikely((ssize_t)(count|iv->iov_len) < 0))
- return -EINVAL;
- if (access_ok(VERIFY_WRITE, iv->iov_base, iv->iov_len))
- continue;
- if (seg == 0)
- return -EFAULT;
- nr_segs = seg;
- count -= iv->iov_len; /* This segment is no good */
- break;
- }
+ count = generic_segment_checks(nr_segs, iov, VERIFY_WRITE);

/* coalesce the iovecs and go direct-to-BIO for O_DIRECT */
if (filp->f_flags & O_DIRECT) {
@@ -2225,31 +2239,11 @@ __generic_file_aio_write_nolock(struct k
 size_t ocount; /* original count */
 size_t count; /* after file limit checks */
 struct inode *inode = mapping->host;
- unsigned long seg;
 loff_t pos;
 ssize_t written;
 ssize_t err;

- ocount = 0;
- for (seg = 0; seg < nr_segs; seg++) {
- const struct iovec *iv = &iov[seg];
-
- /*
-  * If any segment has a negative length, or the cumulative
-  * length ever wraps negative then return -EINVAL.

```

```
- */
- ocount += iv->iov_len;
- if (unlikely((ssize_t)(ocount|iv->iov_len) < 0))
- return -EINVAL;
- if (access_ok(VERIFY_READ, iv->iov_base, iv->iov_len))
- continue;
- if (seg == 0)
- return -EFAULT;
- nr_segs = seg;
- ocount -= iv->iov_len; /* This segment is no good */
- break;
- }
-
+ ocount = generic_segment_checks(nr_segs, iov, VERIFY_READ);
  count = ocount;
  pos = *ppos;
```

Subject: Re: [PATH][RFC] mm: Move common segments checks to separate function

Posted by [Dmitriy Monakhov](#) on Thu, 25 Jan 2007 06:49:22 GMT

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

Dmitriy Monakhov <dmonakhov@openvz.org> writes:

> Move common segments checks from __generic_file_aio_{read,write}_nolock()
> to separate helper function generic_segment_checks().

WOW my eyes

I'm really sorry but this patch is little bit broken. :(

I've forgot to check generic_segment_checks() return value.

So following is the right version:

LOG:

Move common segments checks from __generic_file_aio_{read,write}_nolock()
to separate helper function generic_segment_checks().

Signed-off-by: Dmitriy Monakhov <dmonakhov@openvz.org>

--- mm/filemap.c.orig 2007-01-25 09:33:03.000000000 +0300

+++ mm/filemap.c 2007-01-25 09:34:25.000000000 +0300

@@ -1126,6 +1126,38 @@ success:

return size;

}

+/*

+ * Performs necessary segments checks.

```

+ * Returns bytes count or appropriate error code that caller should return.
+ */
+static ssize_t
+generic_segment_checks(unsigned long nr_segs, const struct iovec *iov,
+ unsigned long access_flag)
+{
+ size_t bcount;
+ unsigned long seg;
+ bcount = 0;
+ for (seg = 0; seg < nr_segs; seg++) {
+ const struct iovec *iv = &iov[seg];
+
+ /*
+ * If any segment has a negative length, or the cumulative
+ * length ever wraps negative then return -EINVAL.
+ */
+ bcount += iv->iov_len;
+ if (unlikely((ssize_t)(bcount|iv->iov_len) < 0))
+ return -EINVAL;
+ if (access_ok(access_flag, iv->iov_base, iv->iov_len))
+ continue;
+ if (seg == 0)
+ return -EFAULT;
+ nr_segs = seg;
+ bcount -= iv->iov_len; /* This segment is no good */
+ break;
+ }
+ return bcount;
+}
+
+/**
+ * generic_file_aio_read - generic filesystem read routine
+ * @iocb: kernel I/O control block
+ @@ -1146,25 +1178,9 @@ generic_file_aio_read(struct kiocb *iocb
+ size_t count;
+ loff_t *ppos = &iocb->ki_pos;
+
+ - count = 0;
+ - for (seg = 0; seg < nr_segs; seg++) {
+ - const struct iovec *iv = &iov[seg];
+ -
+ - /*
+ - * If any segment has a negative length, or the cumulative
+ - * length ever wraps negative then return -EINVAL.
+ - */
+ - count += iv->iov_len;
+ - if (unlikely((ssize_t)(count|iv->iov_len) < 0))
+ - return -EINVAL;

```

```

- if (access_ok(VERIFY_WRITE, iv->iov_base, iv->iov_len))
- continue;
- if (seg == 0)
- return -EFAULT;
- nr_segs = seg;
- count -= iv->iov_len; /* This segment is no good */
- break;
- }
+ count = generic_segment_checks(nr_segs, iov, VERIFY_WRITE);
+ if (count < 0)
+ return count;

/* coalesce the iovecs and go direct-to-BIO for O_DIRECT */
if (filp->f_flags & O_DIRECT) {
@@ -2225,31 +2241,13 @@ __generic_file_aio_write_nolock(struct k
size_t ocount; /* original count */
size_t count; /* after file limit checks */
struct inode *inode = mapping->host;
- unsigned long seg;
loff_t pos;
ssize_t written;
ssize_t err;

- ocount = 0;
- for (seg = 0; seg < nr_segs; seg++) {
- const struct iovec *iv = &iov[seg];
-
- /*
- * If any segment has a negative length, or the cumulative
- * length ever wraps negative then return -EINVAL.
- */
- ocount += iv->iov_len;
- if (unlikely((ssize_t)(ocount|iv->iov_len) < 0))
- return -EINVAL;
- if (access_ok(VERIFY_READ, iv->iov_base, iv->iov_len))
- continue;
- if (seg == 0)
- return -EFAULT;
- nr_segs = seg;
- ocount -= iv->iov_len; /* This segment is no good */
- break;
- }
-
+ ocount = generic_segment_checks(nr_segs, iov, VERIFY_READ);
+ if (ocount < 0)
+ return ocount;
count = ocount;
pos = *ppos;

```
