
Subject: [PATCH 1/2] mm: move common segment checks to separate helper function (v6)

Posted by [Dmitriy Monakhov](#) on Mon, 12 Mar 2007 07:57:36 GMT

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

I really don't want to be annoying by sending this patchset over and over again. If anyone thinks this patch is really cappy, please comment what exactly is bad. Thank you.

Changes:

- patch was split in two patches.
- comments added. I think now it is clearly describes things.
- make `generic_segment_checks()` inline
- patch prepared against 2.6.20-mm3

How is tested:

- LTP test, and other `readv/writev` op tests.

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

mm/filemap.c | 76 ++++++-----
1 files changed, 39 insertions(+), 37 deletions(-)

diff --git a/mm/filemap.c b/mm/filemap.c

index 8e1849a..0aadf5f 100644

--- a/mm/filemap.c

+++ b/mm/filemap.c

@@ -1159,6 +1159,39 @@ success:

return size;

}

+/

+ * Performs necessary checks before doing a write

+ *

+ * Adjust number of segments and amount of bytes to write.

+ * Returns appropriate error code that caller should return or

+ * zero in case that write should be allowed.

+ */

+inline int generic_segment_checks(const struct iovec *iov,

+ unsigned long *nr_segs, size_t *count,

+ unsigned long access_flags)

+{

+ unsigned long seg;

+ 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(access_flags, 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;
+ }
+ return 0;
+}
+
/**
 * generic_file_aio_read - generic filesystem read routine
 * @iocb: kernel I/O control block
@@ -1180,24 +1213,9 @@ generic_file_aio_read(struct kiocb *iocb, const struct iovec *iov,
    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;
- }
+ retval = generic_segment_checks(iov, &nr_segs, &count, VERIFY_WRITE);
+ if (retval)
+     return retval;

    /* coalesce the iovecs and go direct-to-BIO for O_DIRECT */
    if (filp->f_flags & O_DIRECT) {
@@ -2094,30 +2112,14 @@ __generic_file_aio_write_nolock(struct kiocb *iocb, const struct

```

```

iovec *iov,
    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;
- }
+ err = generic_segment_checks(iovec, &nr_segs, &ocount, VERIFY_READ);
+ if (err)
+     return err;

    count = ocount;
    pos = *ppos;
--
1.5.0.1

```

Subject: Re: [PATCH 1/2] mm: move common segment checks to separate helper function (v6)

Posted by [Christoph Hellwig](#) on Mon, 12 Mar 2007 08:27:19 GMT

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

On Mon, Mar 12, 2007 at 10:57:53AM +0300, Dmitriy Monakhov wrote:

```

> +/*
> + * Performs necessary checks before doing a write
> + *
> + * Adjust number of segments and amount of bytes to write.

```

```
> + * Returns appropriate error code that caller should return or
> + * zero in case that write should be allowed.
> + */
> +inline int generic_segment_checks(const struct iovec *iov,
> + unsigned long *nr_segs, size_t *count,
> + unsigned long access_flags)
```

Why isn't this static? Also I doubt that we want to inline this function.

Subject: Re: [PATCH 1/2] mm: move common segment checks to separate helper function (v6)

Posted by [Nick Piggin](#) on Mon, 12 Mar 2007 08:31:10 GMT

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

On Mon, Mar 12, 2007 at 10:57:53AM +0300, Dmitry Monakhov wrote:

```
> I really don't want to be annoying by sending this patcheset over and over
> again. If anyone think this patch is really cappy, please comment what
> exactly is bad. Thank you.
```

Doesn't seem like a bad idea.

```
>
> Changes:
> - patch was split in two patches.
> - comments added. I think now it is clearly describe things.
> - make generic_segment_checks() inline
> - patch prepared against 2.6.20-mm3
>
> How is tested:
> - LTP test, and other readv/writev op tests.
>
> Signed-off-by: Monakhov Dmitry <dmonakhov@openvz.org>
> ---
> mm/filemap.c | 76 ++++++-----
> 1 files changed, 39 insertions(+), 37 deletions(-)
>
> diff --git a/mm/filemap.c b/mm/filemap.c
> index 8e1849a..0aadf5f 100644
> --- a/mm/filemap.c
> +++ b/mm/filemap.c
> @@ -1159,6 +1159,39 @@ success:
>     return size;
> }
>
> +/*
> + * Performs necessary checks before doing a write
> + *
```

```

> + * Adjust number of segments and amount of bytes to write.
> + * Returns appropriate error code that caller should return or
> + * zero in case that write should be allowed.
> + */
> + inline int generic_segment_checks(const struct iovec *iov,
> + unsigned long *nr_segs, size_t *count,
> + unsigned long access_flags)

```

Make it static and not inline, and the compiler will work it out.

This function name doesn't really imply that it returns you the nr_segs and count, but that's not a big deal I guess.

You also don't say that nr_segs should be initialised to the amount you which to write, while count must be initialised to zero.

```

> + {
> + unsigned long seg;
> + 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(access_flags, iv->iov_base, iv->iov_len))
> + continue;

```

Why now insert the above test, and put the below statements inside the branch? OTOH, that makes it less obviously c&p from the others. Maybe a subsequent patch.

```

> + if (seg == 0)
> + return -EFAULT;
> + *nr_segs = seg;
> + *count -= iv->iov_len; /* This segment is no good */
> + break;
> + }

```

You could assign to *count here, once, and remove the requirement that the caller initialised it to zero?

```

> + return 0;
> + }

```

```

> +
> /**
>  * generic_file_aio_read - generic filesystem read routine
>  * @iocb: kernel I/O control block
> @@ -1180,24 +1213,9 @@ generic_file_aio_read(struct kiocb *iocb, const struct iovec *iov,
>  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;
> - }
> + retval = generic_segment_checks(iov, &nr_segs, &count, VERIFY_WRITE);
> + if (retval)
> + return retval;
>
>  /* coalesce the iovecs and go direct-to-BIO for O_DIRECT */
>  if (filp->f_flags & O_DIRECT) {
> @@ -2094,30 +2112,14 @@ __generic_file_aio_write_nolock(struct kiocb *iocb, const struct
iovec *iov,
>  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;
> - }
> + err = generic_segment_checks(iov, &nr_segs, &ocount, VERIFY_READ);
> + if (err)
> + return err;
>
> count = ocount;
> pos = *ppos;
> --
> 1.5.0.1
>

```

Subject: Re: [PATCH 1/2] mm: move common segment checks to separate helper function (v6)

Posted by [Dmitriy Monakhov](#) on Mon, 12 Mar 2007 18:32:18 GMT

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

Nick Piggin <npiggin@suse.de> writes:

```

> On Mon, Mar 12, 2007 at 10:57:53AM +0300, Dmitriy Monakhov wrote:
>> I really don't want to be annoying by sending this patcheset over and over
>> again. If anyone think this patch is really cappy, please comment what
>> exactly is bad. Thank you.
>
> Doesn't seem like a bad idea.
>
>>
>> Changes:
>> - patch was split in two patches.

>> +/*
>> + * Performs necessary checks before doing a write
>> + *
>> + * Adjust number of segments and amount of bytes to write.
>> + * Returns appropriate error code that caller should return or
>> + * zero in case that write should be allowed.

```

```

>> + */
>> +inline int generic_segment_checks(const struct iovec *iov,
>> + unsigned long *nr_segs, size_t *count,
>> + unsigned long access_flags)
>
> Make it static and not inline, and the compiler will work it out.
Wow i've just carefully checked and found more functions with duplicating code:
fs/xfs/linux-2.6/xfs_lrw.c:655 xfs_write()
fs/ntfs/file.c:2339 ntfs_file_aio_write_nolock()
So i think nobody will object against exporting generic_segment_checks()
and removing duplicating code.
>
> This function name doesn't really imply that it returns you the
> nr_segs and count, but that's not a big deal I guess.
>
> You also don't say that nr_segs should be initialised to the amount
> you wish to write, while count must be initialised to zero.
>
>> +{
>> + unsigned long  seg;
>> + 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->iiov_len;
>> +  if (unlikely((ssize_t)(*count|iv->iiov_len) < 0))
>> +  return -EINVAL;
>> +  if (access_ok(access_flags, iv->iiov_base, iv->iiov_len))
>> +  continue;
>
> Why now insert the above test, and put the below statements inside the
> branch? OTOH, that makes it less obviously c&p from the others. Maybe
> a subsequent patch.
>
>> + if (seg == 0)
>> +  return -EFAULT;
>> + *nr_segs = seg;
>> + *count -= iv->iiov_len; /* This segment is no good */
>> + break;
>> + }
>
>
> You could assign to *count here, once, and remove the requirement
> that the caller initialised it to zero?
>

```



```

>> + return 0;
>> +}
>> +
>> /**
>>  * generic_file_aio_read - generic filesystem read routine
>>  * @iocb: kernel I/O control block
>> @@ -1180,24 +1213,9 @@ generic_file_aio_read(struct kiocb *iocb, const struct iovec *iov,
>>  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;
>> - }
>> + retval = generic_segment_checks(iov, &nr_segs, &count, VERIFY_WRITE);
>> + if (retval)
>> + return retval;
>>
>>  /* coalesce the iovecs and go direct-to-BIO for O_DIRECT */
>>  if (filp->f_flags & O_DIRECT) {
>> @@ -2094,30 +2112,14 @@ __generic_file_aio_write_nolock(struct kiocb *iocb, const struct
>>  iovec *iov,
>>  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;
>> - }
>> + err = generic_segment_checks(iov, &nr_segs, &ocount, VERIFY_READ);
>> + if (err)
>> + return err;
>>
>> count = ocount;
>> pos = *ppos;
>> --
>> 1.5.0.1
>>
> -
> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html
> Please read the FAQ at http://www.tux.org/lkml/

```

Subject: Re: [PATCH 1/2] mm: move common segment checks to separate helper function (v6)

Posted by [Christoph Hellwig](#) on Thu, 15 Mar 2007 21:52:33 GMT

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

Actually thinking about this a little more I don't think we should put this in. Instead this loop should be moved up to fs/read_write.c because these are checks that we want for all filesystems/drivers that use vectored I/O. We'll still need tiny loops to calculate the total I/O length for now in every method, but that will be gone as soon as we have our iodesc/uiowhatever structure.
