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 realy don't want to be annoying by sending this patcheset over and over
>> again. If anyone think this patch is realy cappy, please comment what
>> exectly 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 doplicating 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 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
>>
```