Subject: [PATCH 1/2] mm: move common segment checks to separate helper function (v7)
Posted by Dmitriy Monakhov on Mon, 12 Mar 2007 20:19:31 GMT
View Forum Message <> Reply to Message

Changes against v6
 - remove duplicated code from xfs,ntfs
 - export generic_segment_checks, because it used by xfs,nfs now.
 - change arguments initialization pocily according to Nick's comments.

Tested with: ltp readv/writev tests

Signed-off-by: Monakhov Dmitriy <dmonakhov@openvz.org>
---
 fs/ntfs/file.c            |   21 ++---------
 fs/xfs/linux-2.6/xfs_lrw.c |   22 ++----------
 include/linux/fs.h        |    3 ++
 mm/filemap.c              |   83 ++++++++++++++++++++++++++++++++-------------
 4 files changed, 55 insertions(+), 74 deletions(-)

diff --git a/fs/ntfs/file.c b/fs/ntfs/file.c
index dbbac55..621de36 100644
--- a/fs/ntfs/file.c
+++ b/fs/ntfs/file.c
@@ -2129,28 +2129,13 @@ static ssize_t ntfs_file_aio_write_nolock(struct kiocb *iocb,
  struct address_space *mapping = file->f_mapping;
  struct inode *inode = mapping->host;
  loff_t pos;
- unsigned long seg;
  size_t count;  /* after file limit checks */
  ssize_t written, err;

  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_READ, iv->iov_base, iv->iov_len))
-   continue;
-  if (!seg)
-   return -EFAULT;
-  nr_segs = seg;
-  count -= iv->iov_len; /* This segment is no good */

```
-  break;
- }
+ err = generic_segment_checks(iov, &nr_segs, &count, VERIFY_READ);
+ if (err)
+  return err;
  pos = *ppos;
  vfs_check_frozen(inode->i_sb, SB_FREEZE_WRITE);
  /* We can write back this queue in page reclaim. */
diff --git a/fs/xfs/linux-2.6/xfs_lrw.c b/fs/xfs/linux-2.6/xfs_lrw.c
index ff8d64e..558076d 100644
--- a/fs/xfs/linux-2.6/xfs_lrw.c
+++ b/fs/xfs/linux-2.6/xfs_lrw.c
@@ -639,7 +639,6 @@ xfs_write(
  xfs_fsize_t  isize, new_size;
  xfs_iocore_t  *io;
  bhv_vnode_t  *vp;
- unsigned long  seg;
  int   iolock;
  int   eventsent = 0;
  bhv_vrwlock_t  locktype;
@@ -652,24 +651,9 @@ xfs_write(
  vp = BHV_TO_VNODE(bdp);
  xip = XFS_BHVTOI(bdp);

- for (seg = 0; seg < segs; seg++) {
-  const struct iovec *iv = &iovp[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;
-  segs = seg;
-  ocount -= iv->iov_len;  /* This segment is no good */
-  break;
- }
+ error = generic_segment_checks(iovp, &segs, &ocount, VERIFY_READ);
+ if (error)
+  return error;

  count = ocount;
  pos = *offset;
```

```
diff --git a/include/linux/fs.h b/include/linux/fs.h
index 6a3d22e..3b99450 100644
--- a/include/linux/fs.h
+++ b/include/linux/fs.h
@@ -1778,6 +1778,9 @@ extern ssize_t generic_file_sendfile(struct file *, loff_t *, size_t,
read_actor
 extern void do_generic_mapping_read(struct address_space *mapping,
         struct file_ra_state *, struct file *,
         loff_t *, read_descriptor_t *, read_actor_t);
+extern int generic_segment_checks(const struct iovec *iov,
+  unsigned long *nr_segs, size_t *count,
+  unsigned long access_flags);

 /* fs/splice.c */
 extern ssize_t generic_file_splice_read(struct file *, loff_t *,
diff --git a/mm/filemap.c b/mm/filemap.c
index 8e1849a..8bd1ea4 100644
--- a/mm/filemap.c
+++ b/mm/filemap.c
@@ -1159,6 +1159,46 @@ success:
  return size;
 }

+/*
+ * Performs necessary checks before doing a write
+ * @iov: io vector request
+ * @nr_segs: number of segments in the iovec
+ * @count: number of bytes to write
+ * @access_flags: type of access: %VERIFY_READ or %VERIFY_WRITE
+ *
+ * Adjust number of segments and amount of bytes to write (nr_segs should be
+ * properly initialized first). Returns appropriate error code that caller
+ * should return or zero in case that write should be allowed.
+ */
+int generic_segment_checks(const struct iovec *iov,
+   unsigned long *nr_segs, size_t *count,
+   unsigned long access_flags)
+{
+ unsigned long   seg;
+ size_t cnt = 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.
+   */
+  cnt += iv->iov_len;
```

```
+  if (unlikely((ssize_t)(cnt|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;
+  cnt -= iv->iov_len; /* This segment is no good */
+  break;
+ }
+ *count = cnt;
+ return 0;
+}
+EXPORT_SYMBOL(generic_segment_checks);
+
 /**
  * generic_file_aio_read - generic filesystem read routine
  * @iocb: kernel I/O control block
@@ -1180,24 +1220,9 @@ 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 +2119,14 @@ __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
```