

NFS: NFS3 – page null fill in a short read situation

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-06/4041.html>

From: Linda Dunaphant (linda.dunaphant_at_ccur.com)

Date: 06/17/05

To: trond.myklebust@fys.uio.no

Date: Thu, 16 Jun 2005 21:19:51 -0400

Hi Trond,

One of our applications running on a Linux client experiences partial file data corruption (nulls) if the NFS filesystem on a non–Linux server is mounted as NFS3. It works properly if mounted NFS2. The Linux NFS client creates a file on the server and issues the following sequence of lseeks, writes, and read calls:

init write buffer to non–null values

lseek: SEEK_SET to 52

write: 224 bytes

lseek: SEEK_SET to 4096

write: 224 bytes

lseek: SEEK_SET to 0

read: 276 bytes

Using ethereal, I found that the data read back over the wire from the server for bytes 0 to 276 was correct (bytes 52–275 were not null). However, the data returned to the application for bytes 52 to 275 was null.

We have several non–Linux NFS servers that do not return the EOF status flag on a read when data is being returned. A second read at the updated offset returns the EOF flag. This causes the `nfs_readpage_result()` short read logic to be used in this case. If some progress is made on the read (`count != 0`), `nfs_readpage_result()` adjusts the starting position in `pgbase` and `offset`, and decrements the `count` by the amount that was completed. It then restarts the read call. Once the original `count` is satisfied or an EOF occurs, `nfs_readpage_result_full()` is called to null fill the end of any page(s) that were not satisfied by the read(s).

I found `nfs_readpage_result_full()` is not taking into account a short read situation may have occurred. In my example, `data->res.count` was 0 after the second read (for bytes 276 to 4095) due to the EOF. Because `req->wb_pgbase` was 0 and `req->wb_bytes` was 4096, `memclear_highpage_flush()`

Linux-Kernel: NFS: NFS3 – page null fill in a short read situation

cleared all 4096 bytes of the page even though 276 bytes had been read by the first short read. This caused bytes 52 to 275 to be clobbered by nulls.

I changed count in `nfs_readpage_result_full()` to be a total of all the reads by adding `data->args.pgbase` to `data->res.count` (count of the last read). I found that `data->args.pgbase` contained the sum of any previous short reads that may have occurred, and 0 otherwise. In my example, bytes 276 to 4095 are now cleared by `memclear_highpage_flush()` instead of bytes 0 to 4095 which fixes the problem for this application.

Cheers!
Linda

The following patch is for 2.6.12-rc6:

```
diff -ura base/fs/nfs/read.c new/fs/nfs/read.c
--- base/fs/nfs/read.c 2005-06-07 16:21:43.000000000 -0400
+++ new/fs/nfs/read.c 2005-06-16 16:59:45.000000000 -0400
@@ -425,7 +425,7 @@
 */
static void nfs_readpage_result_full(struct nfs_read_data *data, int status)
{
- unsigned int count = data->res.count;
+ unsigned int count = data->res.count + data->args.pgbase;

    while (!list_empty(&data->pages)) {
        struct nfs_page *req = nfs_list_entry(data->pages.next);
```

-

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/>