Subject: Problem: LTP linkat01 test fails on nfs directory (NFS v3)
Posted by gblond on Fri, 21 Sep 2007 10:16:33 GMT
View Forum Message <> Reply to Message

Hello.

Tested kernels: 2.6.18, 2.6.22, 2.6.23-rc2

Steps to reproduce: Suppose that we have mounted some directory from nfs v3 server with default options. Also we have the two directories in this mountpoint and each directory has hard linked file. Try to open those files and write to one and read from another. Data will not be equal. (Testcase: attached hardlink_test.c)

Analysis: Although these hard linked files have same nfs_fattr::fileid but have different nfs_fh fhandle. In this case nfs_find_actor() function (fs/nfs/inode.c) returns false during opening each file:

```
nfs_find_actor()
{
...
 if (nfs_compare_fh(NFS_FH(inode), fh))
  return 0;
...
}
```

Therefore for each of hard links new struct inode is allocated. It leads to cache aliasing.

Please explain why nfs_find_actor() function compares file handles?

--
Thanks,
Vitaliy Gusev

File Attachments

1) hardlink_test.c, downloaded 226 times

Subject: Re: Problem: LTP linkat01 test fails on nfs directory (NFS v3)
Posted by Trond Myklebust on Fri, 21 Sep 2007 12:37:07 GMT
View Forum Message <> Reply to Message

On Fri, 2007-09-21 at 13:13 +0400, Vitaliy Gusev wrote:
> Hello.
>
> Tested kernels: 2.6.18, 2.6.22, 2.6.23-rc2
>

> Steps to reproduce: Suppose that we have mounted some directory from nfs v3
> server with default options. Also we have the two directories in this
> mountpoint and each directory has hard linked file. Try to open those files
> and write to one and read from another. Data will not be equal. (Testcase:
> attached hardlink_test.c)

Please retry after re-exporting the filesystem using the highly
recommended "no_subtree_check" option. The default subtree_check option
is broken: it changes the filehandle of the file upon a cross-directory
rename() of that file.

> Analysis: Although these hard linked files have same nfs_fattr::fileid but
> have different nfs_fh fhandle. In this case nfs_find_actor() function
> (fs/nfs/inode.c) returns false during opening each file:
>
> nfs_find_actor()
> {
> ...
> if (nfs_compare_fh(NFS_FH(inode), fh))
>   return 0;
> ...
> }
>
> Therefore for each of hard links new struct inode is allocated. It leads to
> cache aliasing.
>
> Please explain why nfs_find_actor() function compares file handles?

'cos this is the only way to know that two files are the same. fileid is
not always supported by servers: it is an optional NFSv4 attribute, and
on NFSv3, most non-posix filesystems will fake it using something like
iunique().

Comparing filehandles allows you to be certain that two files are the
same if the filehandles are equal. If they are not equal, then that does
not guarantee that the files are different, but then how else are you
going to determine it?

Trond

Subject: Re: Problem: LTP linkat01 test fails on nfs directory (NFS v3)
Posted by gblond on Fri, 21 Sep 2007 14:39:35 GMT
View Forum Message <> Reply to Message

On the Friday 21 September 2007 16:37 Trond Myklebust, wrote:
> On Fri, 2007-09-21 at 13:13 +0400, Vitaliy Gusev wrote:
> > Hello.

> >
> > Tested kernels: 2.6.18, 2.6.22, 2.6.23-rc2
> >
> > Steps to reproduce: Suppose that we have mounted some directory from nfs
> > v3 server with default options. Also we have the two directories in this
> > mountpoint and each directory has hard linked file. Try to open those
> > files and write to one and read from another. Data will not be equal.
> > (Testcase: attached hardlink_test.c)
>
> Please retry after re-exporting the filesystem using the highly
> recommended "no_subtree_check" option. The default subtree_check option
> is broken: it changes the filehandle of the file upon a cross-directory
> rename() of that file.

Ok, problem goes out. Default option depends on nfs-utils release.

> >
> > Please explain why nfs_find_actor() function compares file handles?
>
> 'cos this is the only way to know that two files are the same. fileid is
> not always supported by servers: it is an optional NFSv4 attribute, and

>From RFC3010 (NFSv4) section 2.4.1 :
  "For example, if paths /a/b/c and /a/d/c refer to the same file, the
server SHOULD return the same filehandle for both path names traversals".

For NFSv4 problem should not be shown.

> on NFSv3, most non-posix filesystems will fake it using something like
> iunique().
>
> Comparing filehandles allows you to be certain that two files are the
> same if the filehandles are equal. If they are not equal, then that does
> not guarantee that the files are different, but then how else are you
> going to determine it?
>

--
Thanks,
Vitaliy Gusev