Subject: Re: I forgot the subject - Confusing ShouldStall() method
From: Andreas-Joachim Peters <andreas.joachim.peters@cern.ch>
Date: 01/11/2019, 08:45
To: Steven Murray <Steven.Murray@cern.ch>
CC: Elvin Alin Sindrilaru <elvin.alin.sindrilaru@cern.ch>, Mihai Patrascoiu <mihai.patrascoiu@cern.ch>, "castor-tape-dev (castor tape developments)" <castor-tape-dev@cern.ch>

Normally a stall sends a WAIT(n seconds) to the client, that means the client hangs and retries after n seconds.
The XrdMgmOfs::prepare function has the macro
MAYSTALL
which does that.
However this function should be changed to increase the rate counter by the number of files to prepare and not only one, because it is one prepare call but doing many prepares internally and if you want to limit something like files/s we should increase this number accordingly.

The prepare function then loops over the files and if I remember right, it calls the event command.

The event command also has a
MAYSTALL
which means, that the loop can be interrupted and the stall response can come back to the client.

So it depends where you put the rate limiter, how it behaves.

In any case, the client does not print anything unless you put XrdCl into debug mode .... it just retries after the WAIT period.

Cheers Andreas.

On Fri, Nov 1, 2019 at 5:01 PM Steven Murray <Steven.Murray@cern.ch> wrote:

> Hi Andreas,
>
> Many thanks for your reply. No problem about the explanation.
>
> I have two more questions, one from German and one from me:
>
> 1. German asks what is a client program supposed to do when it is stalled. Should it just display an error message and exit immediately

with an error code or should it freeze for a moment and retry without any user intervention?
2. My question - Is the XrdMgmOfs::prepare() method interfering with the EOS stall mechanism when it calls the XrdMgmOf::FSctl() for each file in a bulk prepare request and then ignores any stall errors from XrdMgmOf::FSctl() by simply logging them and moving onto the next file?

Cheers,

Steve

On 31 Oct 2019, at 22:54, Andreas-Joachim Peters <andreas.joachim.peters@cern.ch> wrote:

Yes,
this logic does not regulate a single method alone. If any triggers, all requests are stalled.
There is a reason to do it like that, but I don't have much time too explain it now.

Cheers Andreas.

On Fri, Nov 1, 2019 at 5:03 AM Steven Murray <Steven.Murray@cern.ch> wrote:

Hi Elvin, Mihai and Andreas,

This is a resend of my previous e-mail, but this time with a subject and the missing fact that "`XrdMGMOfs::prepare()`" calls "`XrdMgmOfs::FSctl()`".

The EOS source code calls the "`ShouldStall()`" method to detect if a user request should be stalled because they have exceeded a rate limit set by the "eos `access`" command.

The following comment says that the "`ShouldStall()`" method takes a parameter called "`function`" which is used to narrow down the check to just the specified function. Looking at the code I conclude that this parameter is completed ignored! I also conclude that no matter where `ShouldStall()` is called from it can stall the caller for any of the active "access" rate limits. Am I correct?

```
[itctabuild02] ~ > vi eos/mgm/XrdMgmOfs.hh
...
1087
//------------------------------------------------------------------------
------
1088   //! Function to test if a client based on the called function and
his
1089   //! identity should be stalled
1090   //!
1091   //! @param function name of the function to check
1092   //! @param accessmode macro generated parameter defining if this
is a reading
1093   //! or writing (namespace modifying) function
```

```
1094   //! @param stalltime returns the time for a stall
1095   //! @param stallmsg returns the message to be displayed to the
user
1096   //!
1097   //! @return true if client should get a stall, otherwise false
1098   //!
1099   //! @note  The stall rules are defined by globals in the Access
object
1100   //! (see Access.cc)
1101
//------------------------------------------------------------------------
------
1102   bool ShouldStall(const char* function, int accessmode,
1103                    eos::common::VirtualIdentity& vid,
1104                    int& stalltime, XrdOucString& stallmsg);
```

I will be very confused if my conclusion is not correct because a call to the "`XrdMgmOfs::FSctl(SFS_FSCTL_PLUGIN, args, error, &lClient)`" method is generating an error where "`error.getErrText()`" is set to the following

```
  "Attention: you are currently hold in this instance and each request
is stalled for 10 seconds ..."
```

and the one only one rate limit that has been created using the "`eos access`" command is for rate limiting "prepare":

```
  [itctabuild02] ~ > sudo eos access ls
  #
.....................................................................
.............
  # Stall Rules ...
  #
.....................................................................
.............
  [ 01 ]              rate:user:*:Prepare => 1
  [itctabuild02] ~ >
```

Cheers,

Steve