Mods to MarFS Fuse for ShardMDS – fine grained namespace scaling

(\*\*\*\*\*\* the security to make this occur requires that we cant use posix as an access method so access to metadata shards needs to be via secure remote file access (iofsl + security).

A further optimization is that we could use the suid bit for the master directory (under which metadata sharded files virtually exist). This suid bit could be set on for the directory if the directory is sharded making the calls to getxattr on the directories to be far less frequent since the suid bit/perms is in the stat structure already.

In the config file for each namespace, there are fields for slavep (or shardp) and slavepnum (shardpnum)

Shardp or slavep is the path to the shard file systems/filesets for this namespace

Shardpnum or slavepnum is the number of shards to split up files across

Init

At init time we should verify the presence of all the namespace and shard FS mounts as well as the hashed space for trash and shards

Expand path info routine changes

* expand path info needs change to return two paths and number of shards
* expand path info can use the existence of the setuid bit on the directory to see if it needs to do an getxattr
* The current path it returns is the path to the mds directory/file given a user supplied path (expandedpath)
  + This is build the way it is currently built by taking config mds/namespace info and turning the user supplied path into the path to the mdfile or mddirectory
  + /marfs/projecta/xxxxx/foo turns into /mdfs2/namespaceforprojecta/xxxxx/foo
* The new path needed to be returned is the path to the potential mds shard to the file/potential file (expandedshardpath)
  + This new path is build as follows:
    - We will only fill this path in if the namespace is marked as sharded in the config file
    - take the expandedpath from the above current method, cut off and split the field into everything before the last string (preexpandedpath) (separated by slash) and everything after (postexpandedpath), get the shard xattr for that directory in preexpandedpath. If that xattr exists and the number of shards is > 0 then put the number of shards in a new return value, expandedshardpathnumshards. Also construct the path to the md sharded file as follows based on shard info for this namespace in the config file, something like /mdfs1.shard1/namespace3/hash of mdfs directory inode/FILENAME where FILENAME is postexpandedpath
* The new expandedshardpathnumshard return value is set to zero unless the getxattr on the preexpandedpath directory yields a number of shards

So the calling routine gets the expandedpath, expandedshardpathnumshard, and expandedshardpath back. In some cases there wont be anything in expandedshard path because there wont be a directory or the xattr doesn’t exist or something like that.

\*\*\*\* classes of fuse call effects \*\*\*\*\*

FILEOP – for the operations that can only be on files

Return from expandpathinfo

* If expandedshardpathnumshard > 0
  + Use the expandedshardpath as the path to the metadata, as that should be valid because the user is referring to a file in directory preexpandedpath and the directory that file is in is sharded
* If expandedshardpathnumshard < 1
  + Use expandedpath as the path to the metadata, as the user is referring to a file in a directory preexpandedpath that is not sharded

DIROP – for the operations that can only be on directories

Return from expandpathinfo

* Just use expandedpath as the path to the metadata, as the user is referring to a director in a directory expandedpath and all directory ops occur in the main name space and not in shards

FROMTO – for the operations that are from and to operations

* Input path
  + Return from expandpathinfo
  + If expandedshardpathnumshard > 0
    - We don’t know if this is a file or a directory we are operating on
      * Stat expandedshardpath
      * If this is a file then the path to use is expandedshardpath and set a flag that the output has to be a file
      * If this is a directory then it’s a mistake, err because no directories live in shards
      * If it doesn’t exist then just continue
  + If expandedshardpathnumshard < 1
    - Well, we still don’t know if it’s a file or a directory yet but we know its not in a shard
      * Stat expandedpath
      * If it’s a directory then path to use is expandedpath and set a flag that the output has to be a directory
      * If it’s a file then path to use is expandedpath and set a flag that the output has to be a file
      * If it doesn’t exist then err – as there is no valid input
* Output path
  + Return from expandpathinfo
  + If output flag is directory then expandedpath is the output path
  + If output flag is a file and output expandedshardpathnumshard > 0 then output path is expandedshardpath
  + If output flag is a file and output expandedshardpathnumshar < 1 then output path is expandedpath

UNKNOWN – for the operations where the routine could be operating on a directory or a file (it is not known at op call time)

Return from expandpathinfo

* If expandedshardpathnumshard > 0
  + We don’t know if this is a file or a directory we are operating on
  + If expandedshardpathnumshard > 0
    - Try your operation on expandedshardpath
    - If succeeds then return
    - If fails then it still might be a directory not in a shard
  + Try your operation on expandedpath
    - If it’s a dir or if it’s a file in a directory that isn’t sharded it should work or be a legitimate error

Below is the resulting considerations for each fuse call

int(\* [getattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a7a4c5d8eaf7179d819618c0cf3f73724) )(const char \*, struct stat \*)

UNKNOWN

int(\* [readlink](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ab4ce6e6d69dfde3ec550f22d932c5633) )(const char \*, char \*, size\_t)

UNKNOWN

int(\* [mknod](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a1465eb2268cec2bb5ed11cb09bbda42f) )(const char \*, mode\_t, dev\_t)

FILEOP

int(\* [mkdir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a0a38aa6ca60e945772d5d21b0c1c8916) )(const char \*, mode\_t)

DIROP

Make shard directories based on config shardpnum

Add xattr with shardpnum from config file for this namespace

\*\*\* add setuid bit to indicate this directory is to be sharded so you know you should look in the xattr for that directory

int(\* [unlink](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a8bf63301a9d6e94311fa10480993801e) )(const char \*)

UNKNOWN

int(\* [rmdir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ac59578d18db12f0142ae1ab6e8812d55) )(const char \*)

DIROP

int(\* [symlink](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ab86022391e56a8ad3211cf754b5b5ebe) )(const char \*, const char \*)

FROMTO

int(\* [rename](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#aa777cbddc91887b117ac414e9a2d3cb5) )(const char \*, const char \*)

FROMTO

Copy the metadata input to output and unlink input.

Directories may have xattrs now, so you need to copy those too

int(\* [link](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a1b234c43e826c6a690d80ea895a17f61) )(const char \*, const char \*)

for now, I think we should not allow link – its pretty complicated to do

int(\* [chmod](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a7e75d299efe3a401e8473af7028e5cc5) )(const char \*, mode\_t)

UNKNOWN

int(\* [chown](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a40421f8a43e903582c49897894f4692d) )(const char \*, uid\_t, gid\_t)

UNKNOWN

int(\* [truncate](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a8efb50b9cd975ba8c4c450248caff6ed) )(const char \*, off\_t)

FILEOP

int(\* [open](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a14b98c3f7ab97cc2ef8f9b1d9dc0709d) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

FILEOP

int(\* [read](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a2a1c6b4ce1845de56863f8b7939501b5) )(const char \*, char \*, size\_t, off\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

uses FD/fuse open info which should contain the proper path to the metadata

int(\* [write](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a897d1ece4b8b04c92d97b97b2dbf9768) )(const char \*, const char \*, size\_t, off\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

uses FD/fuse open info which should contain the proper path to the metadata

int(\* [statfs](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a4e765e29122e7b6b533dc99849a52655) )(const char \*, struct statvfs \*)

UNKNOWN

int(\* [flush](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ad4ec9c309072a92dd82ddb20efa4ab14) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

don’t know

int(\* [release](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#abac8718cdfc1ee273a44831a27393419) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

uses FD/fuse open info which should contain the proper path to the metadata

int(\* [fsync](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a92bdd6f43ba390a54ac360541c56b528) )(const char \*, int, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

don’t know

int(\* [setxattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a988ced7091c2821daa208e6c96d8b598) )(const char \*, const char \*, const char \*, size\_t, int)

UNKNOWN

int(\* [getxattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ae21503c64fe2990c8a599f5ba339a8f2) )(const char \*, const char \*, char \*, size\_t)

UNKNOWN

int(\* [listxattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ab4a9c361ce48406f07d5a08ab03f5de8) )(const char \*, char \*, size\_t)

UNKNOWN

int(\* [removexattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a5e54de801a0e0d7019e4579112ecc477) )(const char \*, const char \*)

UNKNOWN

int(\* [opendir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a1813889bc5e6e0087a936b7abe8b923f) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

DIROP

Getxattr shardpnum of the directory and put in fuse open dir info

Open all the shard directories and put their FD’s in fuse open dir info

int(\* [readdir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a0f634deda31d1e1c42664585ae820076) )(const char \*, void \*, [fuse\_fill\_dir\_t](http://fuse.sourceforge.net/doxygen/fuse_8h.html#ae2a2054f9852fd6020c26a1bcc7f1042), off\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

uses DIRP

in a loop or using threads readdiring the open shard directories

int(\* [releasedir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a729e53d36acc05a7a8985a1a3bbfac1e) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

uses DIRP

close all the shard directories and the namespace directory

shut down threads if needed

int(\* [fsyncdir](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#aba5cc1fe9a63ec152ceb19656f243256) )(const char \*, int, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

don’t know

void \*(\* [init](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#adc6dc71274f185de72217e38d62142c4) )(struct [fuse\_conn\_info](http://fuse.sourceforge.net/doxygen/structfuse__conn__info.html) \*conn)

read up object config/db into tables in memory

use secure method to read any secrets needed for repos requiring secrets and maybe obscure in memory

void(\* [destroy](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ac41d37ab860204fe4bd7612f9fb036c5) )(void \*)

No change

int(\* [access](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a2248db35e200265f7fb9a18348229858) )(const char \*, int)

UNKNOWN

int(\* [create](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a97243e0f9268a96236bc3b6f2bacee17) )(const char \*, mode\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

Not implemented

int(\* [ftruncate](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a1e492882859740f13cbf3344cf963c70) )(const char \*, off\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

Not implemented

int(\* [fgetattr](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a573d79862df591c98e1685225a4cd3a5) )(const char \*, struct stat \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

Not implemented

int(\* [lock](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a1c3fff5cf0c1c2003d117e764b9a76fd) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*, int cmd, struct [flock](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ad5968d566dab370974043fcf4271eb25) \*)

Not implemented

int(\* [utimens](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a79955861cc5eb006954476607ef28944) )(const char \*, const struct timespec tv[2])

UNKNOWN

int(\* [bmap](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ae3f3482e33a0eada0292350d76b82901) )(const char \*, size\_t blocksize, uint64\_t \*idx)

Not implemented

int(\* [ioctl](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a37f0612d67a6b76bf10fe6a71b0e3b5b) )(const char \*, int cmd, void \*arg, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*, unsigned int flags,

void \*data)

Not implemented

int(\* [poll](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a2c02838d30391c09dd5213edc61e106a) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*, struct fuse\_pollhandle \*ph, unsigned \*reventsp)

Not implemented

int(\* [flock](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#ad5968d566dab370974043fcf4271eb25) )(const char \*, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*, int op)

Not implemented

int(\* [fallocate](http://fuse.sourceforge.net/doxygen/structfuse__operations.html#a4fa8203e4bfa71d62c15deb5dffe4867) )(const char \*, int, off\_t, off\_t, struct [fuse\_file\_info](http://fuse.sourceforge.net/doxygen/structfuse__file__info.html) \*)

Not implemented