Mods to MarFS configuration and Fuse for providing a parallel staging area data only file system that allows update in place ( can be used for bad behaving access methods like fuse and things that could be put on top of fuse like HSI and Globus. PFTOOL can have smarts in it to do packing and make well formed objects directly but all other access methods when writing to MarFS can create not well formed objects, like the lack of packing and even N to 1 writing. If you want to provide N to 1 writing and make a place to stage N to 1 and small files that aren’t packed etc. you can use the SEMI-DIRECT repo type. This SEMI-DIRECT repo type is a file system (needs to be a parallel file system if you are going to do N to 1 writing) where MarFS will put whatever size/access method type files. Typically this would be where you point interactive (irepo) traffic at (fuse). The concept is that the metadata for the file is kept in the normal namespace file system but the data for the file is kept in a normal file (parallel if N to 1 writing is needed). The file path/name/attrs will be put into the normal namespace mds file system (or in a metadata shard if using shards). The file data will be put in the SEMI-DIRECT repo file system. This allows multiple name spaces to share a staging area if desired. The name of the data file will be generated roughly the same as an object type repo. There wont be any recovery information put into the file however like there is in an object in the object repo.

Because there are security issues with using a scatter tree to put these files into a mounted file system, we will use iofsl as a remote secure file access and will modify the comm method for iofsl to pass secrets for file operations.

We will try to mimic the DIRECT access method as much as possible because we want this to act like a Direct file system repo but just not be in the mdfs.

The following is the MAR\_datarepo that describes a repo, see below in red for changes in the config file that indicate a SEMI-DIRECT repo

MAR\_datarepo { (one stanza for each data repo) {

|  |  |
| --- | --- |
| name | Name for this repo, this name is used in the namespace table above in the config file and it is also used stored with the file in xattr, so this can not be changed easily. It follows the same rules as deleting a repo in this list. |
| URLprefix | This is a string associated with the repo used to access the repo  Object names will be repo URLprefex/bucket/objid  Or really URLprefex/namespace.repo/objid (which is formed and stored in the MAR\_objid xattr  For a SEMI-DIRECT repo, this is the base directory of the SEMI-DIRECT repo file system (like /staging) |
| updateinplace | Updates in place for files in this repo are allowed. This lets you decide if a file is in a repo that can do update in place then the FUSE and batch programs can allow update in place. If a repo doesn’t allow this easily then you can forbid it. It is probably good practice to not allow this for all repo’s used in a namespace but you don’t have to do that. No update in place means that if you open for write, you have to overwrite the entire file from the beginning. It also means that you cannot truncate the file to any other value than zero. It also means that you cannot open with append and append to the file , (although this capability might be changed at a later date). The software can use update in place for DIRECT as the repomethodinfo (which tells the software to put the file data in the metadata file). (yes/no). This can be changed but it is not recommended.  Just like for a DIRECT repo type, a SEMI-DIRECT repo allows upate in place |
| repomethodinfo | Info about method for accessing the object repo, like S3 or CDMI  DIRECT means (use the metadata file system for the user data)  SEMI-DIRECT (means there will be a separate file system that will hold the file data in files |
| …. The rest of the fields |  |

}

Init

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

Additionally, at init time any SEMI-DIRECT file system should be verified and the hash space (directory structure for holding the files) should be verified.

The hashed space for a SEMI-DIRECT file system should take on the following naming convention:

/repopath/namespace.shard/hash-of-inodes using a scatter tree concept

where repopath comes from the config file URL-Prefix, namespace.shard comes from the namespace in the config file \*\*\* there can be multiple name spaces using this SEMI-DIRECT repo file system, and then the hash-of-inodes is the scheme where the inode of the mdfile in the namespace or namespaceshard is used as the way spread the files across directores in the SEMI-DIRECT repo file system by taking the lower order digits and making a directory structure out of them.

So if the inode of the mdfile for the file was 987654321 then the path to the file would be something like this

/repopath/namespace.shard/1/2/3/4/987654321

\*\*\* the directory structure /repopath/namespace1.0/1…

and /repopath/namespace2.0/1….

Must be precreated and checked at init time in fuse and these directories should be owned by root or a storageadmin user and be the directoreis world read/write/executable. Security is provided by the remote secure file access IOFSL method, so all access to these files is done using that method (that is implied in all this spec)

We should use the file setuid bit on all files that are semi-direct which indicates that the file size and mtime is not correct in the mdfile, you have to getxattr and then go get the file size and mtime from the semi-direct file via stat of that file.

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

expandpath info routine should get the XATTRs for the name if it exists.

Somehow we need to mark in the XATTR that this is a SEMI-DIRECT in the XATTR for files in SEMI-DIRECT, marking the path for how to get to these semi-direct files via iofsl

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

if the setuid bit is turned on for a file, then stat the file, get the xattr for semidirect, go stat the semidirect file and use the mtime and size from the semidirect file and use the mdfile stat info for all other fields

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

UNCHANGED I think

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

if the repo to write to is semi-direct then setuid bit on

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

UNCHANGED

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

if it’s a semidirect file, unlink the semi-direct file and then unlink the mdfile

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

UNCHANGED

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

UNCHANGED I think

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

UNCHANGED

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)

unchanged, just do the op on the mdfile as the semi-direct file is protected by secure file access

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

unchanged, just do the op on the mdfile as the semi-direct file is protected by secure file access

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

if semi-direct (setuid) then get xattr and truncate the semi-direct file, we never look at the size or mtime of the mdfile on a semi-direct file, we always look at the semi-direct for that info.

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) \*)

If this is a SEMI-DIRECT (setuid) we should have the data file path from expandpath info, so we would open the SEMI-DIRECT file in whatever mode is called for. We also open the mdfile for consistency.

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 and the FD is the data file,

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 and the FD is the data file

The write should just write the data at the offset required into the data file, this means the file size and mtime will be wrong on the mdfile – and we never use that info for semi-direct files

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

UNCHANGED

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 and the FD is the data, so close mdfile and data file

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)

UNCHANGED

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

UNCHANGED

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

UNCHANGED

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

UNCHANGED

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) \*)

UNCHANGED

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) \*)

UNCHANGED

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) \*)

UNCHANGED

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)

UNCHANGED

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])

UNCHANGED

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