GlusterFS APIs

2013-03-08

Bangalore

redhat.

Libgfapi Basics

- Do some things manually that glusterfs[d] does in main()
  - *Create a context*
  - *Load a volfile into that context*
  - *Set up logging, etc.*
- Issue individual calls using glfs_xxx
  - *e.g. glfs_open, glfs_write*

Libgfapi Code Example

```
glfs_t    *fs = NULL;
int        ret = 0;

fs = glfs_new ("iops");
if (!fs) {
        return 1;
}

ret = glfs_set_volfile_server (fs, "tcp", "localhost",
                                   24007);
ret = glfs_set_logging (fs, "/dev/stderr", 7);
ret = glfs_init (fs);

fd = glfs_creat (fs, filename, O_RDWR, 0644);
```

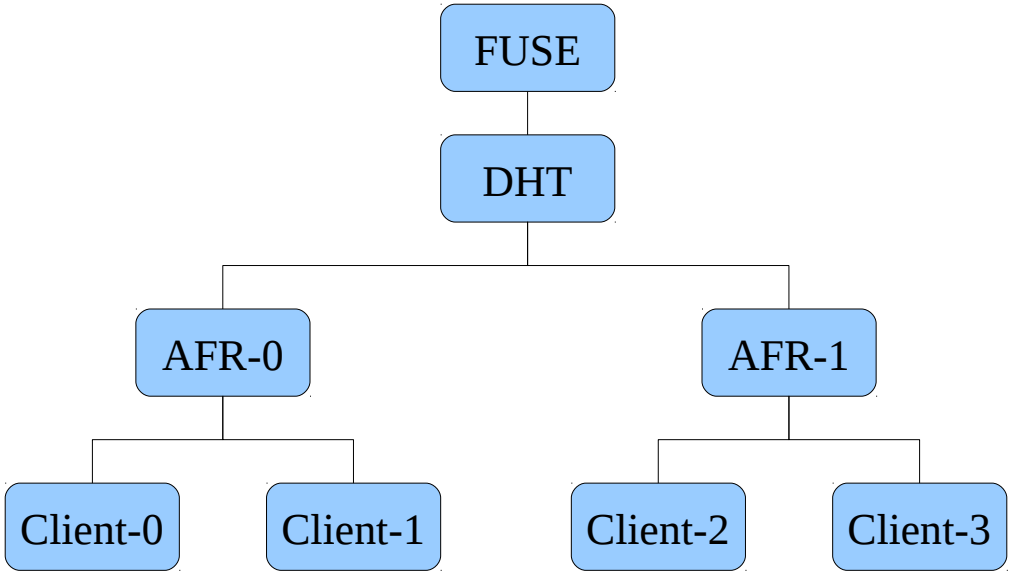
Libgfapi in Python

```
vol = Volume("localhost",volid)
vol.set_logging("/dev/null",7)
vol.mount()

def test_create (vol, mypath, data):
    fd = vol.creat(mypath,os.O_WRONLY|os.O_EXCL,0644)
    if not fd:
        print "creat error"
        return False
    rc = fd.write(data)
    if rc != len(data):
        print "wrote %d/%d bytes" % (rc, len(data))
        return False
    return True
```

Translator Basics

- Add functionality from simple storage “bricks” to user (at tree root)



Translator Environment

- All written in “Plain Old C” - thank you, Dennis
- STACK_WIND to call “down” (toward storage) at the beginning of a request
- STACK_UNWIND to call “up” (toward user) at the end of a request
- General filesystem-related functions
  - *get/set context on inodes and fds, dictionaries, …*
- Utility functions
  - *memory allocation, string handling, logging, …*
  - *more about this later*

Here Be Dragons

- All calls are asynchronous
  - *Continuation Passing Style, node.js, etc.*
- You lose control when you call up or down
- You regain control via callback

```
1267          STACK_WIND (frame, dht_unlink_cbk,
1268                      cached_subvol, cached_subvol->fops->unlink,
1269                      &local->loc);
1270
1271          return 0;
```

callback pointer

world might have changed

Private Context

- Accessed via *frame->local*
- Available to both original dispatch function and callback, so it's very handy
- Can be any structure you want (*void *)*
- You are responsible for allocating it, but freed automatically (if non-null) along with the frame
- Tricky object-lifecycle rules are a recurring theme (e.g. same issues with dictionaries)

Request Types

- Mostly what you'd expect from POSIX or Linux VFS
  - *open, close, readv, writev, stat, …*
  - *… but watch out e.g. for separate lookup function*
- Most operate on file descriptors or inodes
  - *sometimes both, e.g. fstat vs. stat*
- All have request-specific arguments
  - *paths, modes, flags, data (of course)*
  - *hint: use default_xxx* as a template for each *xxx* you implement

Dictionaries

- String-valued key plus arbitrary value
- Used for translator options, and also as arguments for some requests (e.g. *getxattr*)
- Lots of functions to set different kinds of values
  - ints, strings (static/dynamic), binary blobs
- Enumeration via *dict_foreach*
- Both dictionaries (*dict_t*) and data items (*data_t*) have refcounts

Persistent object context

- Inode (*inode_t*) and file descriptor (*fd_t*)
- Treat translator as key, arbitrary value
  - setting value also triggers forget/release callback when the object itself is destroyed

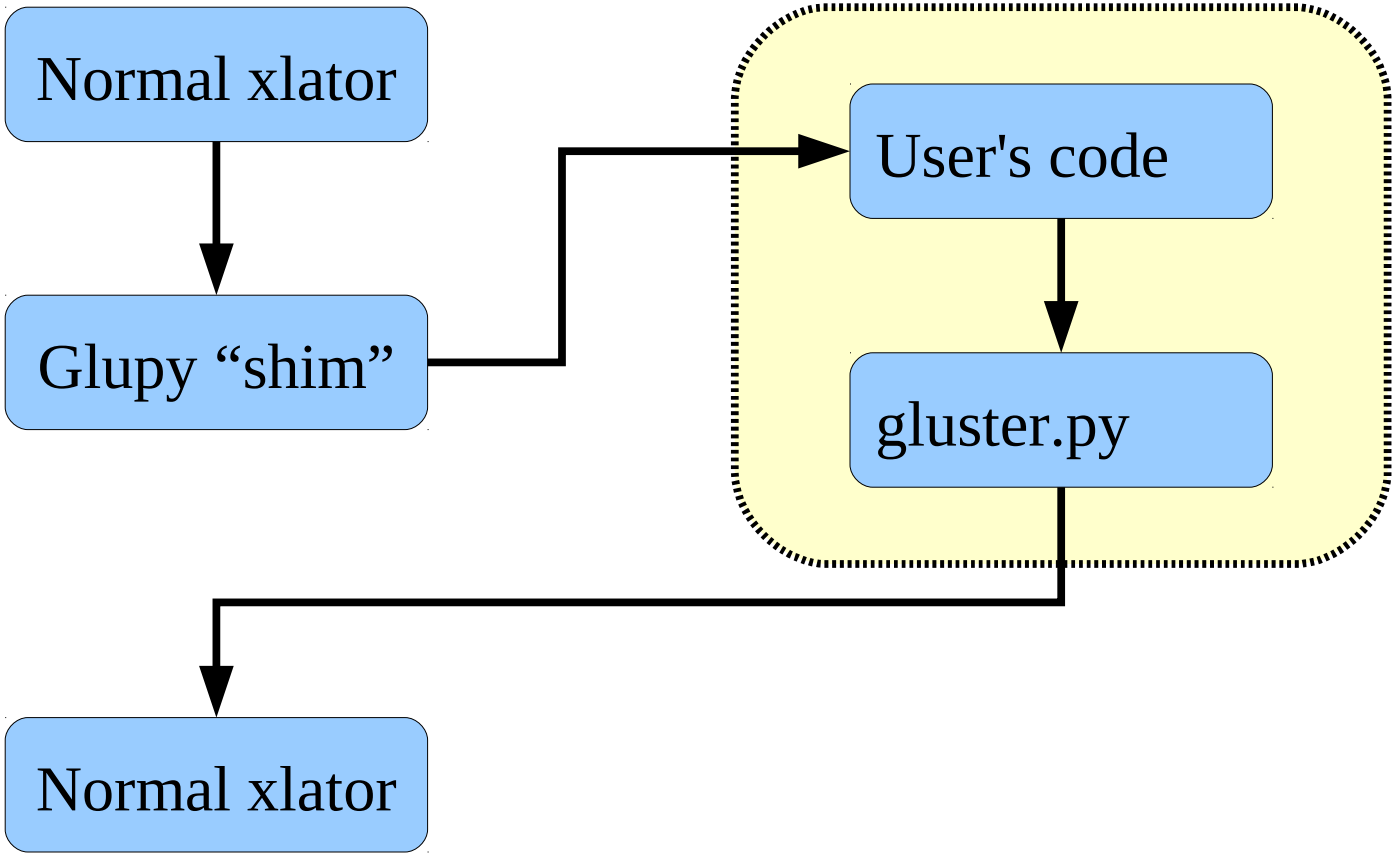
```
inode_ctx_put (inode, xlator, value)
inode_ctx_get (inode, xlator, &value)
inode_ctx_del (inode, xlator, &value)
fd_ctx_set (fd, xlator, value)
fd_ctx_get (fd, xlator, &value)
fd_ctx_del (fd, xlator, &value)
```

Translator Code Example

```
int32_t
negative_lookup (call_frame_t *frame, xlator_t *this, loc_t *loc,
                 dict_t *xdata)
{
         ...

        gf_log(this->name,GF_LOG_DEBUG,"%s/%s => MISS",
               uuid_utoa(loc->gfid), loc->name);
        gp = GF_CALLOC(1,sizeof(ghost_t),gf_negative_mt_ghost);
        if (gp) {
                uuid_copy(gp->gfid,loc->pargfid);
                gp->name = gf_strdup(loc->name);
        }
        STACK_WIND_COOKIE (frame, negative_lookup_cbk, gp,
                           FIRST_CHILD(this),
                           FIRST_CHILD(this)->fops->lookup,
                           loc, xdata);
        return 0;
}
```

Glupy


Normal xlator
Glupy “shim”
User's code
gluster.py
Normal xlator

Glupy Code Example

```
def create_fop (self, frame, this, loc, flags, mode,
                umask, fd, xdata):
    pargfid = uuid2str(loc.contents.pargfid)
    print "create FOP: %s:%s" % (pargfid,
                                 loc.contents.name)
    key = dl.get_id(frame)
    self.requests[key] = (pargfid, loc.contents.name[:])
    dl.wind_create(frame,POINTER(xlator_t)(),
                   loc,flags,mode,umask,fd,xdata)
    return 0
```