15-440 Recitation 5: Intro to DFS Lab cont.

Vijay Vasudevan

Announcements

- DFS Lab Part 1
  - Due tomorrow, October 8 at 11:59pm
- Updated rpctest.cc available on assn page
  - Reduces number of threads in concurrent_test (should run more quickly with RPC_LOSSY=5)
- DFS Lab Part 2, 3 out tomorrow
- Midterm coming up...

Recall: you're building a DFS!

- 4 stages, each building on each other
  1. Lock server, at-most-once RPC semantics
  2. Implement extent server; create/lookup/ readdir FUSE ops
  3. Implement read/write/open/setattr
  4. Implement mkdir/unlink, integrate locks!

Today: Part 2

- Discussing Part 2 today
  - May discuss parts of Parts 3-4 if time

Outline

- Extent server
- FUSE!
- Semantics of filesystem calls

What you've built so far*

- Lock server
  - Can acquire/release arbitrary lock ids
- Augmented RPC framework with at-most-once RPC semantics

*so far = as of 11:59pm tomorrow...

Frangipani

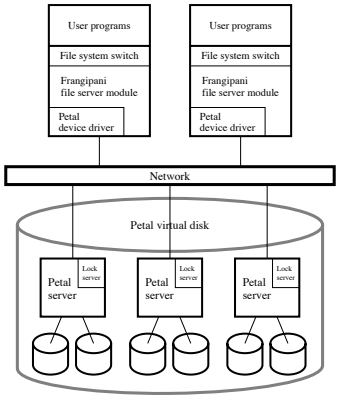
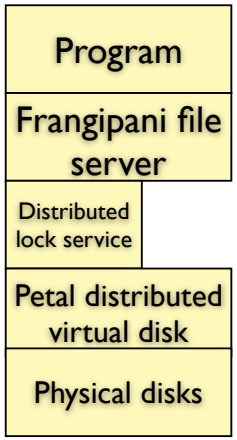
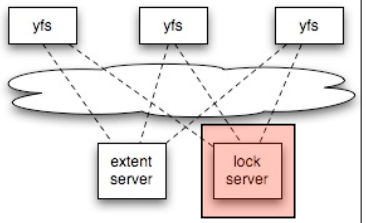


Figure 2: **Frangipani structure.** *In one typical Frangipani configuration, some machines run user programs and the Frangipani file server module; others run Petal and the distributed lock service. In other configurations, the same machines may play both roles.*



YFS

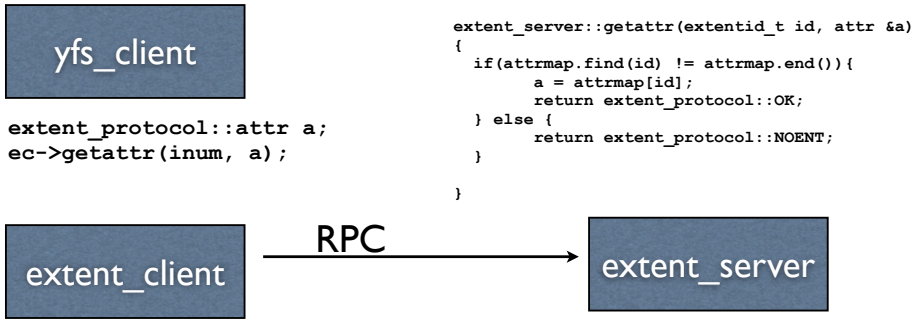
- YFS is much simpler
  - One extent, lock server
  - No "virtual disk"
  - Integrates with FUSE



The extent server

- Don't have to deal with storing data on disk: you will store contents in memory
  - map<inode, std::string>
- All yfs_client operations synchronize with same extent server

example: getattr




```
extent_client::getattr(extentid_t eid,
                       attr &attr)
{
  extent_protocol::status ret = extent_protocol::OK;
  ret = cl->call(extent_protocol::getattr, eid, attr);
  return ret;
}
```


Your job

- Extend extent_server to support
  - put(extentid, string, ...)
  - get(extentid, string&)
  - remove(extentid, ...)
- See extent_protocol.h and extent_smain.cc
  - Later you will likely add more!
- Must properly deal with ctime/atime/mtime

Data formats at extent server

- Directories maintain a mapping of filenames to inode numbers
- if root (inode 1) has two files "file1", "file2":
- get(1) might return

  file1:3983293923

  file2:3384927553

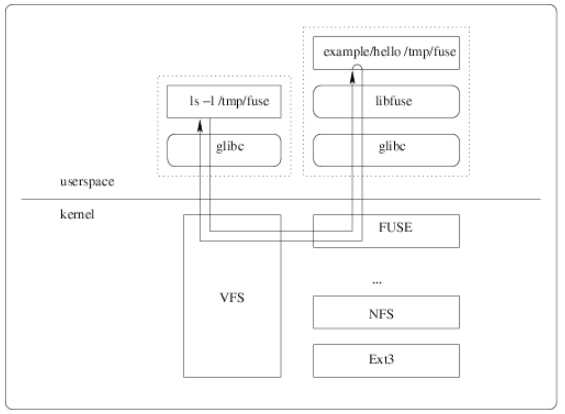
You can choose how you store this information as a string

Metadata time mgmt

- atime: access time
  - Updated whenever file contents accessed
  - Set to 0 on file creation
- mtime: modification time
  - Updated when file contents modified
- ctime: change time
  - Updated whenever metadata modified

FUSE

- Filesystem in Userspace



Mapping FUSE functions

In fuse.cc::main

```
fuseserver_oper.getattr    = fuseserver_getattr;
fuseserver_oper.statfs     = fuseserver_statfs;
fuseserver_oper.readdir    = fuseserver_readdir;
fuseserver_oper.lookup     = fuseserver_lookup;
fuseserver_oper.create     = fuseserver_create;
fuseserver_oper.mknod      = fuseserver_mknod;
```

} you implement these in part 2

```
/* Uncomment these 4 lines for LAB 3 */
//fuseserver_oper.open       = fuseserver_open;
//fuseserver_oper.read       = fuseserver_read;
//fuseserver_oper.write      = fuseserver_write;
//fuseserver_oper.setattr    = fuseserver_setattr;

/* Uncomment these 4 lines for LAB 4 */
//fuseserver_oper.unlink     = fuseserver_unlink;
//fuseserver_oper.mkdir      = fuseserver_mkdir;
```


An example

In fuse.cc

```
void
fuseserver_getattr(fuse_req_t req, fuse_ino_t ino, struct fuse_file_info *fi)
{
    struct stat st;
    yfs_client::inum inum = ino;
    yfs_client::status ret;

    ret = getattr(inum, st);
    if(ret != yfs_client::OK){
      fuse_reply_err(req, ENOENT);
      return;
    }
    fuse_reply_attr(req, &st, 0);
}
```

See http://fuse.sourceforge.net/doxygen/fuse__lowlevel_8h.html for more

CREATE/MKNOD

- Generate inode number (rand())
- This is called on files: make sure inode has MSB set to 1
  - Directories have MSB set to 0
- Create the file at the extent server
- Add the name to inode mapping to parent info at extent server

Lookup

- Given: filename, parent inode
- Look through parent directory list
  - Find inode that maps from filename
  - Getattr on that inode
- Fill in return structure (fuse_entry_param)

Readdir

- Given: inode number for a directory
- Get filename:inode mapping for that dir
- For each filename:inode pair
  - call dirbuf_add(...) function provided

At this point...

- You will be able to create empty files in the root directory and do an ls on the root
- You're not yet storing files...

Testing

- ./start.sh
- ./test-lab-2.pl
- ./stop.sh

- test-lab-2.pl creates a bunch of files, tests whether the files the script created are there

Parts 3 and 4

- Part 3: You implement open, read, write, setattr
- Part 4: You implement mkdir, unlink, and locking
- Will be more straightforward having done part 2, but we'll briefly cover this after the midterm.