15-440 DFS and More

Announcements

- Part 1 Due tonight
- Make sure you copy it to tags/part1 and COMMIT!
- We will grade for completion and correctness.
- We will likely save style grading for the very end.
- Midterm next week!

Today

- Parts 2 and 3
- Homework Review (if we have time – answers should go out soon)
- Questions

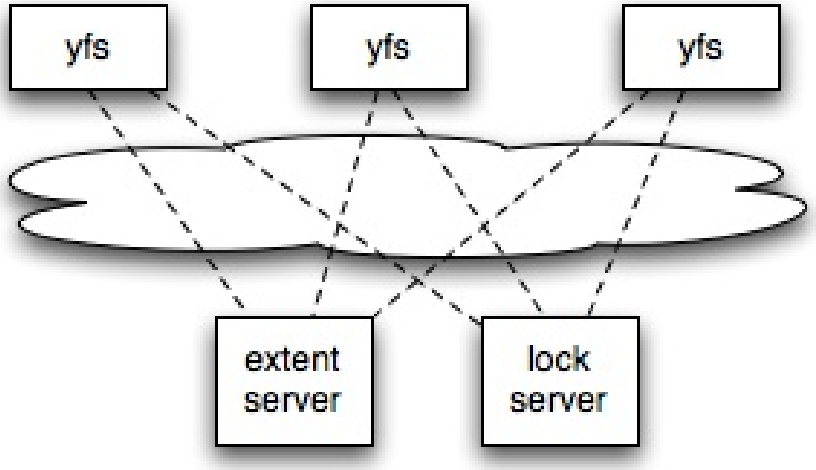
So far

- Lock server – acquire/release arbitrary lock ids
- RPC at-most-once
- Questions?

Next up: Part 2

- Extent Server
- FUSE
- Semantics of Filesystem calls

YFS



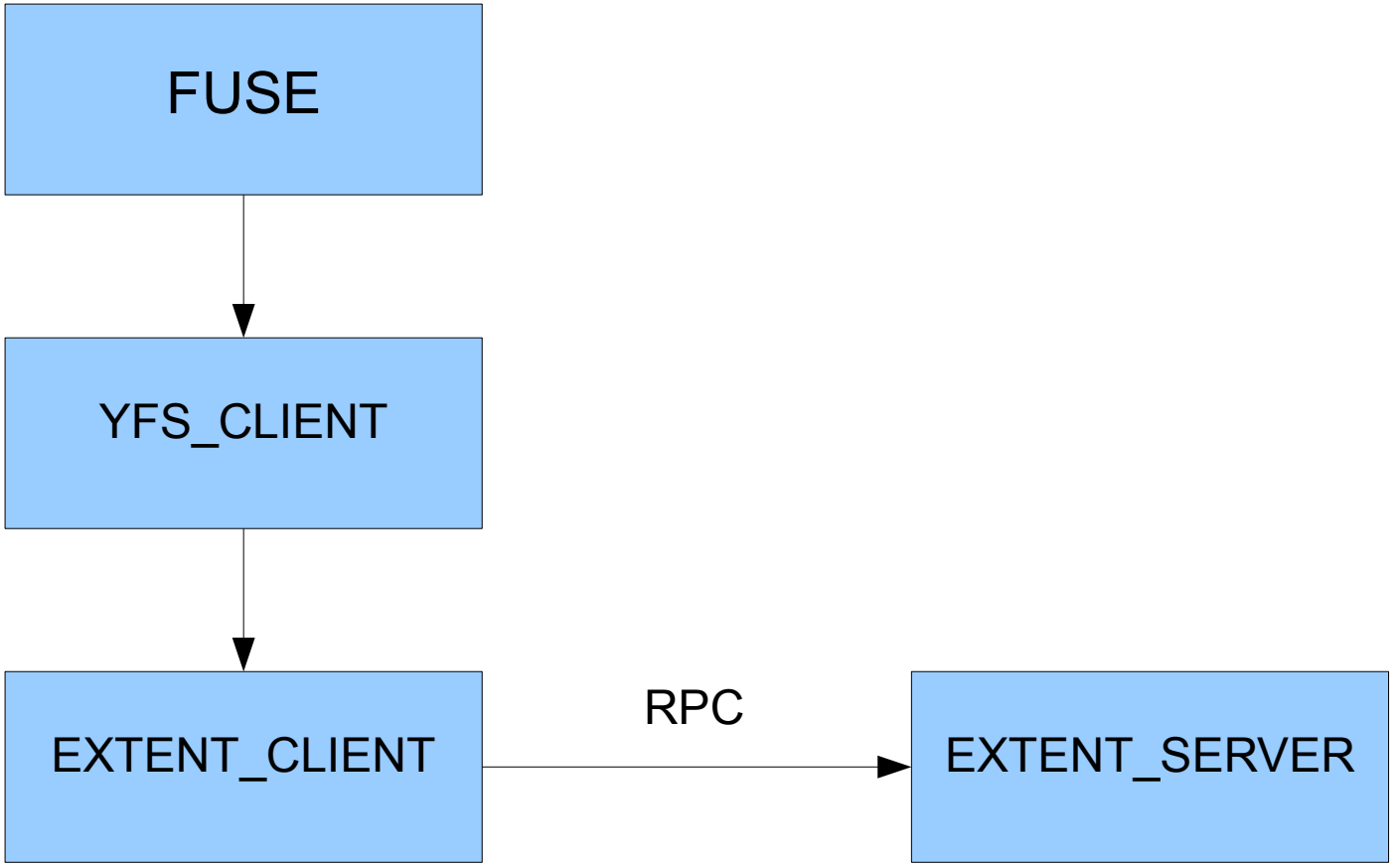
- One extent server
- One lock server
- Integrates with FUSE
- Clients

Extent Server

- You don't have to worry about storing content on disk
- Use a map:

  map<inode, std::string>
- All yfs_clients synchronize with the same extent_server

Structure



Example:getattr

In yfs_client:

```
  if (ec->getattr(inum, a) != extent_protocol::OK) {
    r = IOERR;
    goto release;
  }
```

In extent_client:

```
   ret = cl->class(extent_protocol::getattr, eid, attr);
```

In extent_server:

```
     if(attrmap.find(id) != attrmap.end()){
          a = attrmap[id];
          return extent_protocol::OK;
     }else{
          return extent_protocol::NOENT;
     }
```

Your job

- Extend extent_server to support:
  - put(id, str)
  - get(id)
  - remove(id)
- extent_protocol.h, extent_smain.cc
- ctime/atime/mtime

Data formats

- Directories on the extent server maintain a mapping of filenames to inode numbers
- For example, if the root node (1) contains the files “file1” and “file2”
- The directory entry might look like this:

  file1:2349234

  file2:2408280

  - You get to choose to how to represent this mapping

Metadata

- atime: access time
  - updated when file contents accessed
  - set to 0 at file creation
- mtime: modification time
  - updated when file contents modified
- ctime: change time
  - updated when file metadata modified

FUSE

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;
```

CREATE/MKNOD

- Generated inode number (rand())
- Files have most significant bit set to 1
- Directories have most significant bit set to 0

(later lab)

- Create file at extent_server
- Add the name to inode mapping at the server

Lookup

- Input: filename, parent inode
- Look through parent directory list
- Find the inode that maps from filename
- Call getattr on that inode
- Fill in return structure (fuse_entry_param)

Readdir

- Given inode number for a directory
- Get the list of mappings from that directory
- For each filename:inode pair

  call dirbuf_add(...) //examples in code

At this point

- You should be able to create empty files in the root directory and run ls
- You're not yet storing files (that comes soon)

Testing

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

- test-lab-2.pl creates a bunch of files and then tests whether they are there

Part 3

- Open
- Read
- Write
- Setattr

Open

- OPEN should simply check if the file exists and return fuse_reply_open(req, fi) or an error otherwise
- It doesn't need to fill in any parameters

Read/Write

- Straightforward
- Read:
  fuseserver_read(fuse_req_t req, fuse_ino_t ino, size_t size, off_t off, struct fuse_file_info *fi)

  - It means: read size bytes starting at offset off from file with inode number ino

Setattr

- See handout
- All you have to worry about is the size of the file, but you are welcome to implement other attribute changes if you wish

Questions about the project?

Homework 1 Review