Reddit_FS

Andrew Scott
Harvey Mudd College

Rohin Lohe
Harvey Mudd College

Abstract

We present reddit_fs, a file system that allows users to browse through content from `reddit.com` without an Internet browser. The posts to `reddit.com` and the comments on these posts create a hierarchy similar to that of a tree-structured file system, which we leverage to create the structure of our file system. The file system provides a way for users to view the content of posts and comments by presenting the content as a file with the appropriate extension. For example, an image is represented as a jpg file and the text of a comment is represented as a txt file. `reddit_fs` provides a unique interface for users to browse Reddit without the need for an Internet browser.

1 Introduction

Reddit is an online forum for discussions of different topics. The website is broken up into different "subreddits" dedicated to different topics; for example `https://www.reddit.com/r/filesystems/` is a forum (albeit a relatively inactive one) for file system discussions. Each subreddit consists of posts by users, with the main page for each subreddit being a listing of posts sorted in a particular order according to the user. For example, one possible sort key is "new", which when specified will list the newest posts in chronological order.

A user can make a Reddit account, which will allow them to subscribe to certain subreddits, meaning that posts from those subreddits will show up on their front page when logged in (the front page is a mixture of posts from different subreddits and is what is seen when viewing reddit.com). There are 50 default subreddits that every user is initially subscribed to and which are shown to anyone accessing Reddit while not logged in.

Each post on Reddit takes one of two forms: a "self-post" or a "link-post". A self-post consists of text and is hosted on `reddit.com`, whereas a link-post is a link to a URL outside of Reddit. Each post is voted on by logged-in users. Users can either "upvote" or "downvote" each post, and some of the sort keys that users can specify when viewing a subreddit will sort posts based on user votes. Each post also has a comment section in which users can comment on the post and reply to other comments. The subreddit-post-comment hierarchy mimics the directory tree structure in most modern file systems, which we also mimic with `reddit_fs`.

In the rest of this paper, we will discuss the structure of `reddit_fs` and some challenges that we encountered while building it. We will also discuss future features and improvements that could be added to improve the usability of `reddit_fs`.

2 File System Structure

`reddit_fs` has four primary levels. A full 4-level path in our file system has the following form:

```
/subreddit/sortkey/post/comment
```

After the fourth level, there may be additional levels that are replies to previous comments. These additional levels have the same structure as the comments sublevel.

The following sections describe how each part of the path is displayed and implemented in the file system.

2.1 Subreddits

The first part of a path in `reddit_fs` is the name of a subreddit. This name is identical to the name that appears on `reddit.com` (including capitalization). Once the file system is mounted, all of the logged-in user's subscribed subreddits are loaded using a Python Reddit API Wrapper (PRAW) call [1]. PRAW is a Python package that allows users to easily access the `reddit.com` API. If the user is not logged in, then the 50 default subreddits are loaded. When checking the subreddit portion of a path, we simply check if the given name exists in our loaded subreddits, and when reading the root directory we give back all of the loaded subreddits.

2.2 Sort Keys

The second part of a path in `reddit_fs` is a sort key. Posts in a subreddit can be sorted by one of five methods: new, controversial, top, hot, and rising. Specifying a sort key in the path tells the file system which of the PRAW API calls to use to fetch posts from Reddit when considering the rest of the path. Our implementation is simplified by keeping a list of the sort keys, so when checking the sort key portion of a path the file system can check if the given sort key is in our list and when reading a subreddit directory the file system can give back the contents of our list.

We considered the possibility of Reddit adding a new sort key to the current list. Although our current implementation will not automatically adjust to such a change, we can add any additional sort keys to a list, which will update the file system to handle a new sort key.

2.3 Posts and Comments

The third part of a path is a post title, which is then followed by any comments made in reply to the post. After the first comment, there can be any number of additional replies to comments, and replies to these replies. The depth of these nested comments is reflected identically in our file system. There are two difficulties to deal with when using Reddit post titles as directory names. For one, Reddit post titles have a maximum length of 300 characters [2], which seems a little too long to be reasonable in a file system. Additionally, there is no restriction on two posts having the same title or two comments having the same content, which can cause issues when trying to determine which post or comment a user is requesting with their path. Fortunately, each post has a unique 6-character alphanumeric ID and each comment has a unique 7-character alphanumeric ID [3], both of which can be leveraged to make sure each post has a unique path in the file system. To form the path of a post, our system first truncates the post title or comment body at the last space before 73 characters, and then appends a space and the ID of the post. If there are no spaces in the first 73 characters, then the path is the first 73 characters concatenated with a space and post ID. This ensures that each post will have a path that is unique and a reasonable amount of characters.

3 Content Files

As mentioned before, each post and comment directory contains at least one file representing its content. For a comment or self-post, the content is represented by a single text file containing the title and body of a self-post or the body of a comment (comments don't have titles). For link-posts, there are several possibilities for the type of content file that the file system will give to the user. Our code has special cases to handle several domains, including 3 of the top 4 linked domains linked from Reddit (considering youtube.com and youtu.be to be the same domain) [4].

youtube.com We use a Python library called

PyTube [5] to access the video at the given Youtube link and download the video as an mp4 file. The file shows up in the file system as "content1.mp4".

imgur.com We use a Python library called PyImgur [6] to access the image(s) at the given Imgur link and download each image as either a jpg file or an mp4 file. Imgur links can be to an album of images or to a single image. Each image can either be animated (a gif file) or stationary. If the image is animated, we download it as an mp4 file, and if it is stationary we download it as jpg. We download up to 10 images if we have an album, or the single image if we have one image. The files show up in the file system as "content1.jpg/mp4" up to "content10.jpg/mp4".

streamable.com We use the `streamable.com` API [7] to access the short videos. Streamable's videos are downloaded as an mp4 file and they show up in the file system as "content1.mp4".

gfycat We use the gfycat API [8] to download the linked animated gif file as an mp4 file. The file shows up in the file system as "content1.mp4".

Other domains If the given url is not in any of the domains listed above, we then make a regular HTTP request to the URL using Python's requests library [9]. If this request does not succeed (such as getting a 403 error), we give a text file to the user that lists the URL and says that we were not able to access it.

If we are able to access the URL, we then check the content-type field of the header of the response we get back. If the type is "application/pdf", we write the content of the response we got to a PDF file, which shows up in the file system as "content1.pdf".

If the type of the response is "text/html", we write the content of the response to an html file that shows up in the file system as "content1.html". If the type of the response is neither "application/PDF" nor "text/html", we give the user a text file explaining that our file system doesn't recognize the type of the content.

All of these content files are stored in the underlying file system, which means that if they were all stored as they appear in the file system we would have name overlaps (for example two files with the name "content1.txt". To handle this, we create a unique filename for each file using the unique ID that Reddit provides for each comment and submission, and we keep a mapping from each particular content file to its unique filename. When a user opens a file, the content is stored in the underlying file system with the unique filename, and any read requests to the content file are redirected to that file.

When a user unmounts `reddit_fs`, all of the content files stored in the underlying file system are removed. As a result, when a user remounts `reddit_fs`, nothing from the previous session is saved. Also, since the content files are not removed until the file system is unmounted, the content file corresponding to a post or comment that has been deleted since the file was loaded will still be visible to the user until they unmount the file system.

4 Challenges

We faced several challenges when building the file system. The major ones are described in the following sections.

4.1 MoreComments Objects

Popular posts on Reddit can have many replies to comments, and replies to these replies. The depth of this post–comment tree can reach a depth of 10,000 (although it rarely goes past 10) [10]. Reddit tries to make a nicer user interface by displaying a "More Comments" link that leads to hidden comments, such that users can view a breadth of comments easily. This subset of hidden comments is represented as "MoreComments" objects by the PRAW API. We can access these additional comments with an additional API call (we are not sure of the exact reason for this, but it seems to be done to save space on the client side and to reduce network traffic).

Reddit also limits the total number of comments that will be initially returned to 200,

meaning that there may be a fairly large number of MoreComments objects that need to be "dereferenced" to get all the top level comments on a post [11]. To make things worse, Reddit will give back at most 20 comments at a time when dereferencing a MoreComments object (returning the 20 comments and another MoreComments object), meaning that we could have to dereference a large number of MoreComments objects to get all of the comments that belong in one directory. Since we are limited to 60 requests per minute by Reddit's API guidelines, PRAW inserts a delay of one second between each request to Reddit [3]. As a result, dereferencing a large number of MoreComments objects creates a noticeable lag in the performance of `reddit_fs`. For this reason, we limit the total number of comments at each level in our file system by an adjustable parameter. The larger the number of comments that are allowed at each level, the longer the potential delay from dereferencing a large number of MoreComments objects.

4.2 File Attributes

To make `reddit_fs` perform reasonably well, we have to limit the number of HTTP requests we make. One place where this becomes particularly relevant is when a user asks for the attributes of one of our content files. We offer a semi-true promise that we will not actually download the contents of a content file until it is opened. However, a user can make a `getattr` call on a content file before it has been opened, which is problematic because to know its size we need to make an HTTP request. Unfortunately, in some cases this means that we have to receive a full HTTP response and set the size to be the length of the response's content. However, there are some exceptions for the domains we handle. `imgur.com`, `gfycat.com` and `streamable.com` each require two HTTP requests to actually download content. The first request returns JSON (a simple data exchange format) [12] that contains URLs to jpg or mp4 files that can be downloaded (along with other formats) as well as the size of these files. The second request is then made to one of the URLs from the first request. When we only want the size of the file, we thus only have to make the first request. However, for any of our content files it makes sense to save the content to its backing file if the size of the content is below a certain threshold (if a user asks to open the file later, it would be inefficient to have to make another HTTP request for a small amount of content). This threshold is an adjustable parameter.

Additionally, when a `getattr` call is made on one of our content files, we populate the fields of the stat struct relevant to time (`st_atime`, `st_mtime`, `st_ctime`). The `st_atime` and `st_ctime` fields are filled with the time that the post that the content file represents was made. The `st_mtime` field is filled with the time that the post was last edited, or the time the post was made if the post has not been edited. The edit and post times that we get from PRAW for each post and comment are in the form of a Unix timestamp in local time as opposed to UTC time. In order for the user of `reddit_fs` to see local times, we translate the timestamp to a UTC timestamp using Python's calendar and datetime libraries [13, 14].

4.3 Caching

To minimize the number of HTTP requests we have to make, particularly since we are limited by the PRAW API to 1 request to Reddit per second, we save the results of most of our function calls that require an HTTP request. Currently this is done using a Python dictionary, which will not scale for long periods of use of `reddit_fs`. This can be fixed by using a simple LRU cache, such as `pylru` [15], which would cap the total number of function call results being stored while keeping those values most likely to be used again shortly (this is especially relevant since FUSE has a tendency to make a bunch of FUSE function calls at once).

5 Future Work

There are some features of `reddit_fs` that we wanted to add but did not have time to imple-

ment. Additionally, we were given some suggestions for improvements that could be made which we would implement if given more time.

5.1 Writing Posts and Comments

Users who are logged in may want to write posts and comments on Reddit. To allow this capability, a "newpost" file could be placed in the sort key subdirectory. This location makes the most sense because then the user would have to select a subreddit prior to writing a post on Reddit. Every post and comment directory can have a `newcomment` file. If a user writes to the `newcomment` file, then they are writing a comment that replies to the post in the content file. Since writes are often split into pieces, actually writing a post or comment with PRAW when we get a write to a newpost/newcomment file would probably not make sense. Instead, a more sensible approach would be to wait until the file being written is closed, because then we can get the entire content of the post or comment to write to Reddit.

Additionally, it would make sense for a logged-in user to be able to make edits to their posts and comments. For this to happen, the user would need to have write privileges to content files corresponding to posts and comments that they have written. Conveniently, PRAW allows us to check the author of posts and comments, so checking and giving this privilege would be simple to do. Again, we would wait until a user closes a content file they are editing before writing the edits to Reddit.

5.2 Interacting With Posts and Comments

Users who are logged in may also want the ability to vote on or report posts or comments. This interaction could be implemented in one of two ways.

One option would be to have a file for each interaction. A logged-in user could be given an `upvote`, `downvote` and `report` file in the directory for each post and comment, and could write to one of the files to perform the corresponding action on the post or comment. A reason is required when reporting a post or comment, so a user could write their reason to the report file.

Another option would be to have a single action file in each post or comment directory. A logged in user could write to this file, describing the action that they want to take on the post or comment, such as "upvote" or "report: spam". This solution would be a cleaner way to implement more interactions for users in the future if Reddit adds more ways to interact with posts and comments.

5.3 More Comments

The current solution to the problem of MoreComments objects described in 4.1 is inelegant because it does not allow users a way to view all possible comments on a post. There are two potential ways to improve the accessibility of comments to the user while not compromising performance.

One solution is to represent the MoreComments objects as directories of their own alongside the other comment directories. The contents of these directories would be the comments that are returned by dereferencing the MoreComments object. This structure would allow a way for a user to view all of the comments if they desired, but would limit the system to only having to make one network request per section of comments, avoiding the problem described earlier.

Another solution is to represent the MoreComments object as a file alongside the other comment directories. Whenever a user reads this file, for example by running "cat" on the file, we would add the comments from one of the MoreComments objects to the directory with the other comments. The file would remain in the comment directory as long as there are MoreComments objects left whose comments have not been added to the directory. Again, this solution would prevent the problem of making too many requests at once.

5.4 Imgur Albums

As mentioned earlier, a single Imgur album can have exceedingly large amounts of pictures (i.e. more than 10,000 images). Currently, our file system downloads a set number of images from Imgur. This may not be a fair method for the user, however, because they may want to access a picture that has not been downloaded by the file system.

To make our file system more attractive to users, it would make sense to partition these large albums into a tree structure of smaller directories to allow the user to traverse the album easily. Directories could have a maximum number of entries (either images or more sub directories), such as 25. At each level of the directory tree, the search for the image that a user wants would become narrower, allowing a user to find the image they want much quicker than with a long single list of images. For example, an album with 10,000 images would be split into 20 directories, each representing 400 images. Each of these directories would then be split into directories of 20 or fewer images. If the user knew they wanted image 9,800, then they could simply go to the album titled '9600-10000' and then to the album titled '9800-9816, meaning that they would only need to search through 16 images to find the one they wanted. In addition, each base subdirectory of images would not be downloaded until the user visited it. This process would save us time and possibly the user's Internet connection because we would be downloading a smaller subset of images from the larger album.

There are a few reasons why we chose to display Imgur Albums this way. First, albums on Imgur appear on a single web page. If there are 10,000 images in an album, then the user must scroll to the bottom of the page to view the ten-thousandth image. Our interface gives users a quicker way to view any random image in an album than Imgur's web interface. Second, our design decision was centered on improving our file system's speed. If we had one directory with 10,000 images, then there would be a significant overhead to create the files. The use of several directories to split up the album into smaller partitions allows the file system to make fewer API calls to Imgur. We considered asking Imgur for captions for each image so that we could sort the album alphabetically. The captions would give the user more information about the images (which could make finding a particular image easier). However, a major problem here is that every image does not have a caption. Additionally, some albums only consist of images with no caption. This would defeat the purpose of sorting by caption entirely. Thus, we decided to sort by image number and create small subdirectories in order to decrease the file system overhead.

5.5 Other Content Types

When `reddit_fs` encounters a domain it doesn't recognize, it currently writes the content into either a PDF file or an html file. However, there are other potential content types, such as MP3 files, that we may get from a URL that we currently do not handle. Fortunately the code is flexible and adding cases to handle MP3 files or other content extensions is not difficult.

6 Conclusion

`reddit_fs` is a convenient way to browse Reddit in the form a file system. Users can choose to log in and thereby specify which subreddits are displayed. After selecting a specific subreddit and a sort key, users can navigate through the posts and comments on the web site by navigating through the directories in the file system. A content file placed in each directory appropriately recognizes the post type, and then downloads the contents of the post or comment into the content file. If the post is a link post, the file system is capable of downloading mp4 and jpg files from a few select web sites using their APIs. Otherwise, the content is downloaded in its raw html or PDF form into the content file. If the post is a self post, then the entire text of the post is visible in the content file.

Future work on this project includes adding the ability to write posts and comments to Red-

dit, handling more forms of content, partitioning large Imgur albums for a more user-friendly experience, and displaying all comments under a post rather than selectively showing only a fraction of all the posts.

7 Acknowledgements

We would like to thank Professor Kuenning for all of his design and implementation advice.

References

[1] PRAW Python Reddit API Wrapper. https://praw.readthedocs.org/en/stable/#, March 2016.

[2] Reddit title character limit. https://www.reddit.com/r/ideasfortheadmins/comments/2cteoe/titles_have_a_character_limit_of_300_would_be/, March 2016.

[3] Reddit API Documentation. https://www.reddit.com/dev/api, March 2016.

[4] Reddit though the ages: Most popular domains shared on Reddit from 2007-2015. https://www.reddit.com/r/dataisbeautiful/comments/3mtkmw/reddit_though_the_ages_most_popular_domains/, April 2016.

[5] Nick Ficano. A Python library for downloading YouTube videos. https://github.com/nficano/pytube, April 2016.

[6] Andreas Damgaard Pedersen. The simple way of using Imgur. https://github.com/Damgaard/PyImgur, April 2016.

[7] Streamable API. https://api.streamable.com/, April 2016.

[8] Gfycat API. https://gfycat.com/api, April 2016.

[9] Requests: HTTP for Humans. http://docs.python-requests.org/en/master/, April 2016.

[10] Does Reddit limit the depth of comment nesting at all? https://www.reddit.com/r/Enhancement/comments/1zoly3/does_reddit_limit_the_depth_of_comment_nesting_at/, April 2016.

[11] Comment Parsing. http://praw.readthedocs.org/en/stable/pages/comment_parsing.html, April 2016.

[12] Introducing JSON. http://www.json.org/, April 2016.

[13] Basic date and time types. https://docs.python.org/2/library/datetime.html, April 2016.

[14] General calendar-related functions. https://docs.python.org/2/library/calendar.html, April 2016.

[15] A least recently used (LRU) cache for Python. https://pypi.python.org/pypi/pylru/, April 2016.