Help Topic: The DULoad Toolbox Rsync Wrappers

Spring 2020

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

The DULoad Toolbox Rsync Wrappers

The DULoad Toolbox is designed to facilitate the use of `rsync` in maintaining a remote directory of files. It contains two perl scripts, one for downloading and one for uploading files, which are essentially wrappers for `rsync`. It also utilizes a meta-file on the local machine to facilitate regular transactions with much less complexity than the initial transaction. The `rsync` command is distributed by default in GNU/Linux and MacOS, and is very powerful, but also involves a set of lengthy command-line arguments which can be error-prone. The goals of the DULoad utilities are:

- Hide the complexities of `rsync` to make operations easier to execute and less error-prone.
- Implement key metafiles on both the remote and local machines to facilitate synchronizing sub-trees.
- Implement support for masking out "raw" versions of video and images.

The DULoad package is designed for a user who needs access to a portion of a large data archive, where the whole archive far exceeds the capacity of the user's local disk storage budget. Typically this may be an archive with large amounts of video or image files. The user may want the whole directory structure locally (without files), but may want the ability to selectively populate portions of the directory structure with files of current interest. The DULoad package implements further support for a directory structured with raw and low-resolution versions of media files. This enables users to download low-resolution versions as a preview, and only pull-down higher-resolution files if needed.

Getting the DULoad Package

The DULoad Toolbox consists of two perl scripts: `dload.pl` and `uload.pl`, and a Bash script `rrf.sh`. If you are part of the PavLab group, already with access to the Oceanai server, you can just check out this tree via SVN:

```
$ svn co svn+ssh://oceanai.mit.edu/svn/repos/project-pavlab
```

The directory `project-pavlab/utils/bin` should then be added to your shell path.
For members outside the lab, the DULoad Toolbox scripts can be download in a single tar file from here:

```
$ wget http://oceanai.mit.edu/duload/project-duload.tar
```

Once these scripts are added to your shell path, they're ready to go.

Also see [1].

Basic Usage and Working Example

The duload utilities are designed to be invoked on your local machine, working with another machine acting as a remote server, e.g., the `oceanai` server used in our lab. It is assumed that you have an account on the remote server with the ability to `ssh` onto that machine (preferably with ssh-keys). We'll use the below directory as an example. It is a simple event from which two videos were made. The raw videos are stored, along with a lower-resolution version of each.

```
event_aug1616/
  raw_vids/
    in_water.mov          1.5 GB
    on_shore.mov          1.8 GB
  vids/
    in_water.mp4          181 MB
    on_shore.mp4          472 MB
```

Let's assume it exists on a machine with IP address 18.38.2.158, and on the local file system as `/raiddrive/archives/`. Assuming you have read access on that machine for this directory, you could copy it onto your local machine with:

```
$ scp -rp 18.38.2.158:/raiddrive/archives/event_aug1616 .
```

The drawback of `scp` however is that if this transmission is interupted before completion, a subsequent `scp` invocation would start over again. A better approach with `rsync` is:

```
$ rsync -aP --partial 18.38.2.158:/raiddrive/archives/event_aug1616 .
```

If the above `rsync` transmision were interrupted, the subsequent invocation would only copy files that were not copied on the first transmission. With the `--partial` option, a file interrupted mid-transmission would resume only to copy the remaining part of the file.

So `rsync` is clearly advantageous. The `dload.pl` utility uses `rsync` and implements a few other important features.

Basic Usage of the `dload.pl` utility

Using the `dload.pl` utility, an initial synchronization begins with:

```
$ dload.pl --init=18.38.2.158:/raiddrive/archives/event_aug1616
```

This will copy all the remote directories, but not the files:

```
$ cd event_aug1616
$ ls
raw_vids/  vids/
```

Once the tree has been initialized locally, we can fill it in later at our discretion. Even though we have only downloaded the directories, we can peek at the contents with:

```
$ cd event_aug1616
$ dload.pl --all --raw --list-only
drwxrwxr-x        4096 2017/07/10 16:50:32 .
drwxrwxr-x        4096 2017/07/10 16:50:32 raw_vids
-rw-------  1565214170 2017/07/10 16:48:34 raw_vids/in_water.mov
-rw-------  1911673613 2017/07/10 16:49:33 raw_vids/on_shore.mov
drwxrwxr-x        4096 2017/07/10 16:52:53 vids
-rw-r--r--   189222648 2017/07/10 16:51:25 vids/in_water.mp4
-rw-r--r--   494034665 2017/07/10 16:51:15 vids/on_shore.mp4
```

In this example, we may recall that the `in_water` video may have been interesting. So now that we can see the exact file name, it can be pulled down for preview with:

```
$ cd event_aug1616/vids
$ dload.pl in_water.mp4
```

And if it turns out that the high-resolution version is wanted, it can then be pulled down with:

```
$ cd event_aug1616/raw_vids
$ dload.pl in_water.mov
```

In practice, it is common to download a directory in its entirety *except* all the raw files. By default this is the mode `dload` operates in. *Raw files are all files who's filename or directory name begins with string* `"raw_"`*, case insensitive.* In our example, if the user wanted to download the entire `event_aug1616` directory without raw files:

```
$ cd event_aug1616/
$ dload.pl --all
$ ls *
raw_vids:
vids:
in_water.mp4  on_shore.mp4
```

Using the `--dry-run` Command Line Option

The `rsync` utility supports a `--dry-run` commnand line switch which shows the user which files *would* be transferred if actually invoked without this option. Returning to our example, in this case with raw videos not yet downloaded:

```
$ cd event_aug1616/
$ ls *
raw_vids:                          <-- note raw videos are missing
vids:
in_water.mp4  on_shore.mp4

$ dload.pl --all --raw --dry-run
receiving file list ...
8 files to consider
raw_vids/                          <-- raw videos *would* be downloaded
raw_vids/in_water.mov
raw_vids/on_shore.mov

sent 98 bytes  received 262 bytes  720.00 bytes/sec
total size is 4175680306  speedup is 11599111.96

$ ls *
raw_vids:                          <-- raw videos are still not here
vids:
in_water.mp4  on_shore.mp4
```

The `--list-only` switch only lists the directory contents on the remote machine, and does not show you which remote files are locally missing, and vice versa. The `--dry-run` switch presents you the *differences* between local and remote machines. Using `--dry-run`, or simply `-d`, is a good habit to use prior to most operations, just to confirm that you're actually doing what you think you're doing.

Note: In `rsync` the `--dry-run` option may be abbreviated to `-n`. In the `dload.pl` and `uload.pl` utilities, this is abbreviate to `-d`.

Dowloading Only a Portion of a Directory

If a portion of a remote archive is all that is needed, this portion can be just dowloaded directly with the appropriately extended URL. In our example, if only the `vids` are wanted:

```
 $ dload.pl --init=18.38.2.158:/raiddrive/archives/event_aug1616/vids
 $ cd vids
 $ dload.pl --all
```

(In the rare case that you later decide you want a portion of the tree above the original sub-tree dowloaded, you have three options. (1) The brute force option is to simply start over by downloading the higher level tree. This may be fine if the amount of data is relatively small and bandwidth is high. (2) Download the higher level tree (not yet with files), and move the previously downloaded files into place. (3) Create the higher-level parent directories by hand, move the `.rsync_info` to the root of the tree, and edit this file accordingly.)

Basic Usage of the `uload.pl` utility

The `uload.pl` utility is used for uploading files to tree that has been previously downloaded via `dload.pl`. By example, let's add another video to our local `vids` folder and upload it:

```
$ cd event_aug1616/vids
$ ls
in_water.mp4  on_shore.mp4
$ mv ~/in_air.mp4 .                         <-- move a new video into local folder
$ uload.pl in_air.mp4
Will handle file: in_air.mp4
building file list ...
1 file to consider
in_air.mp4
    15535210 100%    7.00MB/s    0:00:02 (xfer#1, to-check=0/1)

sent 15537234 bytes  received 42 bytes  6214910.40 bytes/sec
total size is 15535210  speedup is 1.00ls
```

The `--all` (or `-a`) switch can be used to upload everything in the local directory (including all local directories recursively) to the remote machine:

```
$ uload.pl -a
```

The `--user=USER` switch can be used if the local username is different than the remote username.

```
$ uload.pl --username=jane
```

Local Storage of the `rsync` Root Location

A key motivation for implementing the `dload.pl` and `uload.pl` scripts is to ease the user burden on getting complex command line arguments correct. We really want to be able to confidently remove any part of the local tree (re-claiming local storage) and get it back with a trivially simple operation. For example:

```
$ cd event_aug1616
$ ls *
  raw_vids:
  in_water.mov  on_shore.mov
  vids:
  in_water.mp4  on_shore.mp4

$ rm -rf raw_vids
$ ls *
  vids:
  in_water.mp4  on_shore.mp4

$ dload.pl -a -r
  raw_vids:
  in_water.mov  on_shore.mov
  vids:
  in_water.mp4  on_shore.mp4
```

Remember the tool is designed for the scenario where the user is working with a portion of an archive that may be too large to hold in its entirety on one's local computer. Easily deleting and restoring parts of the tree is a primary design goal of `dload.pl`.

In the above example, after removing the `raw_vids` directory, `rsync` could have been used directly to restore the tree with

```
$ cd event_aug1616
$ rsync -aP 18.38.2.158:/raiddrive/archives/event_aug1616/raw_vids .
```

Easy, right? You just have to precisely remember the original URL including the server IP address and the location of the files on the remote machine. And, if you accidentally put a trailing `"/"`, as in `../raw_vids/`, then you get the contents of `raw_vids` instead of the directory. The `dload.pl` utility accomplishes the same with:

```
$ cd event_aug1616
$ dload.pl -a -r          (-r is short for --raw, -a is short for --all)
```

To accomplish this simplicity, a bit of behind-the-scenes magic needs to happen. When the (initially empty) tree is first downloaded, a single hidden file, `.rsync_info`, is created in the root of the tree. This file contains the "root" of the original tree, URL and all:

```
$ dload.pl --init=18.38.2.158:/raiddrive/archives/event_aug1616
$ cd event_aug1616
$ cat .rsync_info
18.38.2.158:/raiddrive/archives/event_aug1616
```

Whenever `dload.pl` or `dload.pl` is invoked, the script initially looks up the current full path until it finds the `.rsync_info` file. It uses this information, combined with the full present working directory, to re-constuct the proper `rsync` command line argument. Neither `dload.pl` or `uload.pl` will work without it:

```
$ cd event_aug1616
$ rm -f .rsync_info
$ dload.pl -a
The .rsync_info file was not found. Exiting.
```

References

[1] Michael Benjamin. The duload toolbox scheme. http://oceanai.mit.edu/pavlab/help/duload_scheme, 2017.