EXHIBIT 7

Uploads Last modified Friday, August 7, 2009 at 5:10pm by Jack Pan-che

This node explains our high level upload architecture and how uploads are handled in different environments.

Basic upload flow

1. User navigates to form editfoo.php page on www tier that lets them upload data to facebook. editfoo.php includes lib/distfs/dfs.php and calls distfs_get_upload_server() to get the root http host to use in the form's action field, which tells the browser where to send the POST to. The user selects the items they want to upload and hit submit.

2. Data is posted to uploadfoo.php page on the upload tier that receives the upload POST and stores it on our storage or database.

3. Upload page then redirects to editfoo.php?success=1 or ?success=0 based on the success of the data write that occurred on the upload tier. Other metadata about the write is passed in the get args.

Always make sure to call distfs_get_upload_server(). It determines what environment you are in and generates a url to the proper upload host for you.

NOTE: Don't go to an updev url and start browsing the site. If your code depends on this, it is not following the normal upload pipeline described above and will FAIL IN PRODUCTION.

Environments

Since production uploads happen on two separate tiers, we model this on the dev and intern side as well. This helps guard against people writing code that posts to /uploadfoo.php on their sandbox, which would work in dev if you mounted the volumes on your sandbox, but would fail when your code went to prod.

Environment	*WWW URL*	*Upload URL*	*Host(s)*	*Push stage*	*Releases*
Dev	www.devrsXXX.facebook.com	www.updev004.facebook.com	updev004.snc1	N/A	engshare trunk /var/www
Dev Sandbox	www.$sandbox.devrsXXX.facebook.com	www.$sandbox.updev004.facebook.com	updev004.snc1	N/A	sandbox code from /mnt/devrsXXX/data/...
Latest	www.latest.facebook.com	upload.latest.facebook.com	updev003.snc1	P1	/var/www release
Beta	www.inyour.facebook.com	upload.inyour.facebook.com	updev003.snc1	P1	/var/www release
Prod	www.facebook.com	upload.facebook.com	up.sf2p, up.snc1	P4	/var/www release
Internal uploads	dev,intern	intupload.vip.facebook.com	intup.sf2p	P8	/var/www release, but only serves pages from /var/www /html/intern
ads internal upload		adintupload.vip.facebook.com	updev004/5 /6.snc1	N/A	engshare trunk /var/www

Dev

updev004.snc1 is the dev upload server.

If you hit a non-sandbox dev server, distfs_get_upload_server() will return http://www.updev004.facebook.com, which points at /var/www on updev004 and runs fb95 engshare trunk.

updev004 also supports developer sandboxes. It mounts the devrsXXX:/data dirs, and then distfs_get_upload_server detects when you are in a dev sandbox and in those cases, generates a url of http://www.$user_sandbox_name.updev004.facebook.com.

This runs your sandbox code on updev without any config work or code syncing required on your part.

NOTE: If you have a non-devrs dev server and have hacked /var/www to point to your own code checkout, distfs_get_upload_server will not properly detect your sandbox and you will be sent to www.updev004.facebook.com. This aspect of upload sandboxes is currently broken, but we're working on fixing it.

Latest & Intern

updev003.snc1 is the latest/intern upload server.

It serves upload.inyour.facebook.com and upload.inyournew.facebook.com, and the corresponding hosts for latest as well. It has a custom host.conf that lets it mimic a production upload server with /var/www and /var/www-fb95. It is pushed as part of P1.

Prod

upload.facebook.com balances DNS requests across upload-sf2p and upload-snc1 vips, which are the up.sf2p and up.snc1 pools respectively. These hosts run the latest releases at /var/www and /var/www-fb95, and respond to upload.facebook.com and upload.new.facebook.com.

Internal uploads

intupload.vip.facebook.com takes uploads for internal volumes and serves those files as well. This is used for internal email attachments on cortana and xtools, original copies of upload videos, fb fund proposals, etc. This tier only allows php to be loaded from /var/www/html/intern, if you try to load /profile.php, etc, it won't load, since it uses anything not in /intern as the document root for the served files on the internal volumes.

Why have separate a separate upload tier?

The short answer is that there are too many web hosts and uploads have different resource usage from www pages. By having a separate upload tier, we insulate the two use cases from each other and can scale them separately.

Too many web hosts

Plaintiff's Trial Exhibit
PTX-252
Case No. 08-CV-00862

There's too many web hosts to mount all of our storage volumes on them. A lot of our storage is still on NFS, and trying to reliably mount a NFS volume on thousands of hosts is a losing battle. Once the bulk of our storage is on haystacks, we still have the issue of supporting writes from tens of thousands of client hosts (and a hundred apaches on each host). Constraining the number of nodes sending physical writes helps keep the number of connections down to manageable levels.

Different resource usage

PHP upload receivers don't do any display work, they just receive some data, check it for well-formedness, and then write it to disk and issue a redirect to a success page on the www tier.

Also, since uploads are POSTs of user data to us, their apache processes are longer lived. The scaling of the images also takes a lot of cpu and wall clock.

So we have pages in www that are heavy on display, low in computation, and relatively short lived, compared against upload receivers that don't do any display logic, but do lots of cpu work and are long lived.

Photo servers

We also serve photos from dedicated photo tiers instead of serving them from upload or www. This is so we can tune service of the images since the httpd requirements are very different for taking uploads versus serving images, you'll have many more active requests per second when serving images.

Also, the upload tiers are the only tiers that mount volumes with read/write access. The filers export to all other tiers as read-only, so even if you try the mount the vol rw on a photo host, the filer won't let you. This is to make sure that we only mutate data from authorized tiers, and to guard against issues where bad application code running in www or on someone's random sandbox might delete user images.

Updev setup

See the Updev Setup Guide for instructions on how to setup a new updev host. You won't have to look at this unless you are in ops.

Refreshing vols on an updev host

If you think the updev host is missing a vol, login as root and run '/usr/local/pyface/scripts/photos/distfs-mount-manager -a'. It will print out any volumes it adds or removes and tell you if there are errors.