EXHIBIT 6

Photos Storage Architecture Last modified Thursday, December 14, 2006 at 3:15am by Robert Johnsc

Photos Storage Architecture

Each photo uploaded by a user is stored on disk as several files of different sizes. Files are grouped logically into volumes, which are the basic unit of backend storage. A volume contains a few tens of millions of files, and could be physically stored in different ways. The database record for a photo includes the volume in which its files can be found.

The primary storage for photo files is on several netapps. These are large arrays of disks with heads that serve files over nfs. They appear to the rest of the network as a large amount of storage that can be mounted over nfs, and they store data on multiple disks in a redundant and reliable way. Each netapp holds the data for many volumes.

Users interact with two pools of web servers in front of the netapps: pics and upload. These servers mount the netapps' photo directories over nfs. The pic servers serve requests for existing files, and the upload servers run PHP code to upload new files.

Requests for photo files or uploads are split among the pic and upload pools by a load balancer. The pics pool is subdivided into several groups of servers that each handle a different group of volumes, so that when a volume is unavailable only some of the web servers will get bogged down with nfs timeouts.

There are also some stand-alone servers with copies of sealed (i.e. full) volumes on local or iSCSI drives. These run a custom web server called tarhttpd and are heavily optimized for read-only performance, but are not reliable. A second copy of the volume remains on the netapp as a backup. The load balancer will redirect requests to the netapp if it detects that a tarhttpd server is down.

Content Distribution Network

All of these servers sit behind several 3rd party content distribution networks or CDNs. CDNs act as a very large (many TB) cache, and serve photos to users from many geographic locations. User requests for photos go to the CDN first, and if the CDN doesn't have the file in cache it requests it from our servers. This means that we would get very little benefit from running our own caching software, as it wouldn't have nearly as much storage as the CDN. As of this writing about 5% of requests come back to our servers.

We currently work with three CDNs: Limelight, Akamai, and Panther Express. There is code to distribute the load between these services (or our servers directly) in a way that's granular and easy to change, so we're not too dependent on any one service. Changing CDNs still has some cost because we have to serve a lot more files while we build the cache for the new CDN. We eliminate this problem for profile pictures by splitting requests in such a way that all CDNs have a complete copy of the data. This is feasible for profile pics because there are a relatively small number of them. The code for choosing CDNs is in www/lib/distfs/dfscommon.php, look for "buckets".

Volumes

A volume is a collection of files stored together. It consists of a logical volume and one or more physical volumes. We currently have only one physical volume per logical volume, but this is a configuration choice and not a limitation of the software. The details of these volumes are stored in the photos database on cdb002 in the tables distfs_logical_volumes and distfs_physical_volumes. The logical volume is identified by an id number commonly refered to as the "volume id" and includes a virtual ip where the data for the volume can always be accessed. The physical volume has detailed information on the physical location of the data, basically everything required for a pic or upload server to mount it. The physical volume has an ID number that is often the same as the logical volume ID, but this will not alway be the case. Logical and physical volume ids are independent, and should never be assumed to be the same. The term "volume ID" without mention of logical or physical should be assumed to refer to the logical volume ID.

PHP Interface

This is described in more detail here. The code that knows about physical volumes and the specific location of files is abstracted into a module called distfs, with code in www/lib/distfs/dfscommon.php and www/lib/distfs/dfsclient.php.

Plaintiff's Trial Exhibit
PTX-208
Case No. 08-CV-00862