Case Study: Photo.net

March 20, 2001

What is photo.net?

- An online learning community for amateur and professional photographers
- 90,000 registered users
- 700,000 unique visitors per month
- 8+ million page views per month (3+ per second)
  - Peak rate can be 2-3 times the average
- Bandwidth usage:
  - 2.5 Mbit/sec, average
  - 4-5 Mbit/sec, peak

Photo.net Layered Architecture

- ACS
- AOL Server, Oracle
- SunOS 5.7
- Sun E450
- Shared 10 Mbit/sec network connection (burstable to 100 Mbit/sec)
- Storage Networks Fiber Channel Drives
- All sitting behind an F5 load balancer

Approach to Scaling

- Know your bottlenecks
  - Monitor them carefully
  - Understand what happens when a bottleneck is choking the system
- Anticipate your peaks
  - e.g., Traffic patterns, unique visitors
  - Gracefully deal with peaks
    - e.g., Limit or turn off CPU-intensive features
- Plan ahead

Performance/Bottleneck Monitoring

- Need key performance metrics (also helps detect choking)
  - Local - Load, Bandwidth, Page Requests, …
  - Non-local - Time to first byte, time to load page, page success rate, …
- How do we measure what's going on?
  - WebTrends
  - Keynote
  - Super Monitor
  - Super Watchdog
  - Bandwidth monitor
  - Our end users

ACS

- Modules implemented by a set of scripts with embedded SQL, all under CVS control
- Content stored in a database or in the file system (e.g., photos)
- High degree of collaboration/interactivity
  - Each script can access the database several times (both reads and writes)

ACS, cont.

- User activity tracked behind the scenes (more database reads/writes)
- Key bottlenecks: script interpretation, database access (transactions per second)
  - Write better code, use compilation (adp vs. tcl), caching, and database query optimization

AOL Server

- Full-featured WWW server
- Built-in Tcl and Adp support
- Multi-threaded
- Max threads, max connections, max number of db handles determined at startup
- Key bottlenecks: Lock contention (Tcl data-structures, server log, database handles) and some Tcl commands (regexp on large inputs, ns_adp_parse on nested files)
  - Run multiple instances of AOL Server (need cache consistency at ACS level!)

Oracle

- Full-featured, robust, enterprise-class database
- Connects to AOL Server via a driver
- Multi-threaded - can support hundreds of simultaneous connections
- Key bottlenecks - lock contention on frequently accessed tables
  - Decrease time to access/update tables using caching, RAIDs
  - Adding more CPUs won't speed us up if our bottleneck is lock contention. It could actually slow us down

Sun E450

- Older but reliable server hosted at Exodus
- Runs SunOS 5.7 - a stable, commercial-grade OS
- 4 Gig of RAM, 1 system drive, 4 local mirrored drives, 2 fiber-channel virtual drives
- Shared 10 Mbit/sec network connection (burstable to 100 Mbit/sec)
- Key bottlenecks - RAM, CPU (during peaks), disk bandwidth
  - RAM and CPUs maxed out, use Storage Networks for better disk performance

Performance Improvements

- Move to "three-tier" architecture
  - Third tier is a set of light-weight servers in front of the E450
    - Need lots of RAM to cache mostly static files (e.g., using AFS) and cached Tcl results
  - E450 runs Oracle and manages the database

Performance Improvements, cont.

- Akamaize files (e.g., gifs, photos)
- Replicate the architecture
  - Level of tolerable inconsistency varies across ACS (chat vs. bboard vs. user data)
  - Special merge routines would be needed for good performance
- Use a compiled language like Java instead of interpreted scripts