Web Caching
Outline

- Caching workloads
- Cache deployment scenarios
- Cache consistency
- Hit rates
- Dynamic content

Caching Workloads

- Quantitative trends:
  - Increasing number of web web objects (>>3B)
  - More concentrated access patterns?

- Qualitative Trends
  - P2P workloads
  - Dynamic content
  - Content distribution networks
**Trend: Web / P2P Workloads**

<table>
<thead>
<tr>
<th>Web</th>
<th>P2P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text and images</td>
<td>Audio/Video</td>
</tr>
<tr>
<td>&gt; 3B objects</td>
<td>&lt; 1M objects</td>
</tr>
<tr>
<td>8KB average size</td>
<td>1-100+MB size</td>
</tr>
<tr>
<td>Can be dynamically generated</td>
<td>Never change</td>
</tr>
<tr>
<td>Users expect fast response time</td>
<td>High-latency tolerated</td>
</tr>
</tbody>
</table>

**Dynamic Content**

- Advertising: Doubleclick
  - Delivering targeted advertising
- Commerce: Amazon.com
  - Selling/promotion from a catalogue
  - Buying/transactions
- Publishing: CNN.com
  - Personalization
  - News sites, blogs
- Other categories of dynamic content?
Cache Deployments

Where else?

Cache Deployments
Outline

- Content trends
- Cache deployment scenarios

- Cache consistency
  - Expiration and revalidation
  - Why not callbacks?
  - Data Update Propagation (DUP)
- Hit rates
- Dynamic content

Cache Consistency

- Fresh-enough is good-enough
- One writer, many readers
  - Most content changes slowly wrt # reads

- “Expiration” based cache consistency
  - Expires timestamp on each object
  - Cache revalidates content beyond that time
- Why not AFS style callbacks?
Web vs. AFS

- One writer
- Infrequent updates
- 1K - 1M proxy caches
  - **Client browser caches!**
- WAN mostly

- Expires + revalidation
  - If-Mod-Since
  - Etags for more accurate revalidation

Do AFS-style callbacks ever make sense for the web?

Data Update Propagation (DUP)

- IBM’s Olympic Games website
  - Backend: database, content generation
  - Frontend: web caches (accelerators)
- Cache consistency:
  - Manager has callback list with <cache,object>
  - Developer annotates objects with data dependencies (database items)
- Graph of data->object dependencies
  - Invalidate caches and proactive regeneration of objects
  - Why not invalidate & demand-regenerate?
Internet Cache Consistency

- Number of caches
- Network locality
- Frequency of updates
- Integration with web server / publishing:
  - Administrative
  - Technical
- Impact of revalidation
  - Network traffic
  - Business impact (CDN)
- Standard HTTP Expires:
  - Client caches
  - Proxy caches
- Middle-ground:
  - CDNs (Akamai)
- Tight integration:
  - Accelerators (DUP)

Outline

- Content trends
- Cache deployment scenarios

- Cache consistency
- Hit rates
  - Zipf distribution
  - Hit rate drivers
- Dynamic content
Cache Hit Rates

- Object hit rate: reduced latency
- Byte hit rate: reduced bandwidth
- Byte HR < Object HR
  - Byte HR ~ 50-60%
  - Object HR ~ 60-80%
- Cache operator cares about Byte HR
- End-users care about Object HR
  - Conflict between cache operators and users?

Zipf Distribution

- Top 1% objects:
  - 20-35% of accesses
- Top 25-40% objects
  - 70% of accesses
- Few “very popular” objs
- Heavy tail, lots of references go to unpopular objects
Zipf Cumulative

Hit Rate vs. Population

How to increase population size?
Cache Hierarchies

- Larger client population, larger hit rate
- Aggregate client populations with cache hierarchies
  - Misses from local cache go to a parent cache
- Hit rate grows slowly above a few 1000 users
- Each layer adds latency (and bandwidth)

Hit Rate Drivers

- Client population
- Popularity distribution
- Fraction of dynamic content
- Rate of object change
- Cache size vs. working set

Web content:
- Dynamic
- Popularity: $\alpha=0.80$

P2P workloads:
- Huge objects
- Not dynamic
- Latency ok
- Popularity ($\alpha$):
  - Audio: 0.48
  - Video: 0.86
  - Overall: 0.62

Can caching work for P2P?
Outline

- Content trends
- Cache deployment scenarios

- Cache consistency
- Hit rates
- Dynamic content
  - Active-caching
  - Advertising, commerce, publishing
  - Delta-coding

Active-Caching

- Java CacheApplet cached with each object
- On each request, cache invokes applet to:
  - Generate reply, use cached copy, trigger revalidation
  - Maintains on-cache persistent state
- Cache monitors applet CPU and storage use
  - Can “evict” applet and revert to “Expires” consistency
- Does this solve any real problems?
  - Advertising? Commerce? Personalization?
Advertising: Cache-Busting

- Advertisers want to track and target
  - Per-user cookies
  - Per-user, per-pageview ad selection
  - Caches defeat these goals
  - Caches help deliver ad content quickly

- Real-world solution:
  - Redirect for ad selection/logging
  - Cacheable ad image files

- Would active-caching work?

Advertising Databases

- Ad information:
  - “Inventory”, $, targeting criteria (eligible content types, desired user types, time of day), ...

- Placement information:
  - Stats about the traffic to different pages, $, content topic, expected mix of users with different criteria, ...

- Per-user information:
  - Topics of interest, links to registration/marketing profiles, detailed recent ad viewing history, ...

- Business rules for combining the above in real-time

- How to distribute these databases to caches?
**Commerce**

<table>
<thead>
<tr>
<th>Selling</th>
<th>Purchasing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product DB</td>
<td>User DB</td>
</tr>
<tr>
<td>- Inventory, descriptions, promotions, $, cross-selling info, ...</td>
<td>- Credit card, shipping address, ...</td>
</tr>
<tr>
<td>User DB</td>
<td>Transaction system</td>
</tr>
<tr>
<td>- Purchase history, recent browsing, ...</td>
<td>- Credit card clearance, integration to shipping, ...</td>
</tr>
<tr>
<td>Business rules</td>
<td></td>
</tr>
</tbody>
</table>

Distributed databases for this?
Privacy, proprietary concerns?

**Personalized Publishing**

- my.yahoo.com, slashdot.org
  - Personalized pages from sharable/cacheable components
  - Different layouts/subsets/orders sorts

Seems more tractable
- Per-user preferences database must be distributed

- Active-caching? Other cache-side method? Better done at the client w/XML?
Delta-Coding

- Server sends “diffs” against cached copy of page
  - Can reduce bandwidth for dynamic content
  - Still requires round-trip latency to server
- Exploits redundant data already in cache to compress updated object
  - Straight compression (orthogonal)
  - Spring/Wetherall use of Manber fingerprints?

Dynamic Content vs Cache Deployments

<table>
<thead>
<tr>
<th></th>
<th>Dynamic: Advertising</th>
<th>Dynamic: Commerce</th>
<th>Dynamic: Publishing</th>
<th>Static</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Cache</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proxy Cache</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accelerator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dynamic Content vs Cache Deployments

<table>
<thead>
<tr>
<th></th>
<th>Dynamic: Advertising</th>
<th>Dynamic: Commerce</th>
<th>Dynamic: Publishing</th>
<th>Static</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Cache</td>
<td>Delta-coding</td>
<td>Delta-coding</td>
<td>Delta-coding</td>
<td>Delta-coding</td>
</tr>
<tr>
<td>Proxy Cache</td>
<td>Delta-coding</td>
<td>Delta-coding</td>
<td>Active-caching/Delta-coding</td>
<td>Delta-coding</td>
</tr>
<tr>
<td>CDN</td>
<td>Custom (DoubleClick)</td>
<td>Custom (Amazon?, Yahoo Merchants)</td>
<td>Custom (Akamai)</td>
<td>Custom (Akamai)</td>
</tr>
<tr>
<td>Accelerator</td>
<td>Webserver / DUP</td>
<td>Webserver / DUP</td>
<td>Webserver / DUP</td>
<td>Expires</td>
</tr>
</tbody>
</table>

Open Questions

- Distributed databases to push dynamic content into the network
  - Security/privacy of data and rules
  - R/W data and transactions

- Caches for P2P/media workloads
  - Challenges: huge files, long transfers
  - Advantages: static content, fewer popular objects (video), users tolerate latency