Application Layer: Web Proxies and Caches

Srinidhi Varadarajan
Cache Replacement Policies

● Motivation: Caches on Web proxies are of fixed size. Need a mechanism (replacement policy) to remove entries when the cache gets full.

● Popular Replacement Policies
  – Least Recently Used: More common. Derivatives: LRU-MIN, LRU-THRESHOLD
  – Least Frequently Used

● Additional Attributes:
  – Cache distance from server, page content, document size.
Cache Validation

● Problem: If a web page is updated on a server, how do we invalidate any cached copies?

● Solutions:
  – Server (web page designer?) specified Time-To-Live (TTL)
  – Server invalidation: Push mechanism
  – Invalidation Contracts:
Time-To-Live

- Each URL has a server specified Time-To-Live
- Proxies cache pages till the TTL is reached.
- After the TTL expires, proxies issues a conditional get to retrieve the URL.
  - Recall: Conditional get returns a URL’s contents only if it has been updated.
Server Invalidation

- When a page is updated at the server, it sends an invalidate message to all caches.
- Caches can then re-request the page.

- Problems:
  - Requires server to keep track of proxies.
  - Cache update messages from the server will consume a lot of network bandwidth.
Invalidation Contract

- Motivation: Improve performance of the server invalidation algorithm by reducing the number of caches that need to be updated.
- Each URL has an invalidation contract, similar in principle to the TTL mechanism.
- When a URL is updated at the server, the server sends an invalidation message to only those caches whose invalidation contracts have not expired.
- This solution reduces the number of cache invalidation messages. Only caches that access the URL frequently are notified.
Problems

- Single point of failure
  - Failure of the proxy increases perceived user latency during web browsing. Each request still goes through the proxy

- Cache miss causes the proxy to contact the origin server, an expensive operation
Solutions

- **Cache hierarchies**
  - Proxies are interconnected in a tree-like hierarchical structure

- **Distributed Caching**
  - A set of caches act as a single cache
  - Each cache has a subset of the data. Other caches use hints to determine the remote owner of the data.
Cache Hierarchy: Harvest

- Caches are organized in a static tree hierarchy
  - On a cache miss, a proxy cache sends a UDP query to its siblings, parents and the echo port of the origin server
  - The proxy retrieves the object from the fastest site to return a hit.
  - Note: Origin Server always returns a hit
Harvest (cont.)

- If all caches miss:
  - If the origin server is slower than the fastest parent, the proxy retrieves the object through the parent. The parent in turn will repeat the above query process with its siblings.
    - Motivation: Allow cache updates to propagate through all levels of the cache hierarchy.
  - If the response from the origin server is nearly as fast as the fastest parent, retrieve the object directly from the origin server.

- Harvest’s algorithm is used in several popular proxy implementations including Squid.

- Reference:
  - Harvest: A Scalable Customizable Discovery and Access System.
    ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/
Internet Cache Protocol

- Derived directly from Harvest’s hierarchical cache algorithm
- Uses IP multicast to minimize number of messages:
  - On a miss, a proxy cache sends a query to a multicast address. All neighbor caches subscribe to the multicast group. Replies to the query are always sent in unicast mode.
- Current version:
  - ICP version 2
  - Reference: RFC 2186, RFC 2187
Summary Cache

- ICP’s use of UDP queries to neighbors and waiting for their response (or timeout) is expensive.
- What does the query achieve?
  - It asks if a neighbor has a copy of a requested object.
- Same result can be achieved if we know the contents of caches of all neighbors.
  - Sounds inherently expensive
Summary Cache

- Each proxy maintains a summary of the cache directory of all its neighbors.
  - It uses this summary to send a directed query to the cache that has a copy of the requested object.
  - Summary is not necessarily current. There may be a small chance of a false positive.

- To be useful, the summary must be much smaller than the size of the cache directory entry.
  - Can’t use URL, too long
  - Can’t use server name, too ambiguous
  - Uses Bloom filters to create a bitmap representation of the entry.
Errors

- **False Negatives:**
  - A proxy assumes that its neighbor does not have an object when it does
  - Causes unnecessary retrieval from the origin server

- **False Positives:**
  - A proxy assumes that its neighbor has an object when it does not
  - Causes an extra message and object retrieval from origin server

- Target proxy has a stale copy of the requested object.
Bloom Filters

- Original Paper:

- A multiple-hash based probabilistic mechanism for discovering set membership
- Doesn’t generate false negatives.
- Small probability of false positives. The probability can be reduced
Bloom Filters

- **Uses:**
  - A vector \( v \) of \( m \) bits
  - \( k \) independent hash functions.
    Range 1-m
  - For each element \( a \), compute hash functions \( h_1(a), h_2(a) \ldots h_k(a) \)
  - Set the corresponding bits to 1

- **Note:** A bit in the resulting vector may be set to 1 multiple times
Bloom Filters

- To query for existence of an entry b, compute \( h_1(b), h_2(b) \ldots h_k(b) \) and check if the bits at the corresponding locations are 1.
- If not, b is definitely not a member
- Otherwise there may be a false positive. The probability of a false positive can be reduced by k and m
- False positives can also be minimized by maintaining a bit counter at each location that counts multiple 1’s at the same location.
- Each proxy maintains its own local bloom filter and uses it to represent cached document URLs.
Bloom Filters

- Each proxy sends its bloom filter bit array and the hash function spec to all other caches.
- Updates to the cache directory can be sent by sending the entire bit array or just the changes.
- Bloom filters present an excellent trade-off between memory requirements and a small penalty for false positives.
Summary Cache

- **Updates:**
  - Sending bitmap updates for each change is expensive.
  - Use delayed updates and send a new summary (bitmap) only when new documents have reached a threshold.

- **Performance Gain:**
  - reduces the number of inter-cache protocol messages by a factor of 40 to 65,
  - reduces the bandwidth consumption by over 50%
  - eliminates 75% and 95% of the protocol CPU overhead
  - Maintains the same cache hit ratio
Popular Proxies

- Harvest (outdated, but introduced ICP)
- Squid (public domain descendant of Harvest, used by most ISPs worldwide)
- Cache3D (commercialized version of Harvest, owned by NetApp)
- Netscape Proxy Server (department level proxy; comes with SuiteSpot)
- Microsoft Proxy (department level proxy, and semi-firewall; comes with BackOffice)
- CERN Proxy
Reading

Required

  • www.cs.wisc.edu/~cao/papers/share.ps.Z

Recommended

– Harvest: A Scalable Customizable Discovery and Access System.
  • ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/

– Adaptive Web Caching
  • http://irl.cs.ucla.edu/AWC/