Web Caching

COS 316: Principles of Computer System Design
Lecture 9

Amit Levy & Jennifer Rexford

Downloading a Web Page

User visits https://www.youtube.com
Downloading a Web Page (https://www.youtube.com)

Multiple Problems

• User latency
  • Round-trips to query multiple DNS servers
  • Multiple round-trips with the Web server
  • Delivery of a (possibly large) Web item

• Server overhead
  • Handling many requests from many clients
  • Financial costs to deploy enough servers

• Network bandwidth
  • Traffic on many links in multiple networks
  • Financial costs for the affected networks
Caching to the Rescue: Domain Name System

• What to cache?
  • Mapping of popular names to IP addresses
    • E.g., www.youtube.com → 142.251.41.14
  • Mapping of parts of names to DNS server IPs
    • E.g., .com top-level domain → 192.26.92.30

• Where to cache?
  • Local DNS server (e.g., for the campus)
  • Client machine (e.g., user’s browser)

• How to avoid stale information?
  • Cached entries have a limited “time to live”

Caching to the Rescue: Communication Channel

• End-to-end communication
  • TLS: confidentiality, integrity, and authenticity
  • TCP: ordered, reliable delivery of byte stream

• Establishing the channel is expensive
  • Communication delays, creating data structures, and computing keys

• Exploit temporal locality by reusing the channels
Caching to the Rescue: Web Items

- Cache Web items closer to the client
  - Reduce latency
  - Reduce server overhead
  - Reduce use of network bandwidth

Web Caching: Outline

- Cache replacement
  - Popularity distributions
  - Replacement algorithms
- Cache consistency
  - Dynamic items
  - Cache validation
- Cache placement
  - Client’s web browser
  - Client’s network
  - Server’s network
  - Third party (CDN)
- Content Distribution Network
Cache Replacement

Web Caching Should Work Well!

Zipf Distribution
Item of rank $k$ has frequency $\sim 1/k^\alpha$
Web Cache Hit

On cache hit, retrieve the object from the cache!

Web Cache Miss

If I want to store X, what do I get rid of to make space?
Cache Replacement Algorithms

• Which object to evict?
  • Least likely to be used again soon
  • Least expensive to fetch again

• Example algorithms
  • First in first out (FIFO)
  • Least recently used (LRU)
  • Least frequently used (LFU)

• (Note: all fully associative today)

Cache Replacement: First-In-First-Out (FIFO)

• Evict objects added to cache longest ago
• Very simple!

• Three-item cache example:
  • Request stream: a, b, a, c, a, d, a, e, a, f, g

• Can we do better?
Least Recently Used (LRU)

- Evict object used longest ago
  - “Objects used more recently are more likely to be accessed again”
  - Exploits temporal locality

- Implementation: Update access time for every hit

- Three-item cache example:
  - Request stream: a, b, a, c, a, d, a, e, a, f, g
  - Request stream: h, h, h, i, j, k, h

Least Frequently Used (LFU)

- Evict object with fewest hits
  - “Objects used more often are more likely to be accessed again”
  - If tie, use LRU

- Implementation: Update access count for every hit

- Three-item cache example:
  - Request stream: a, b, a, c, a, d, a, e, a, f, g
  - Request stream: h, h, h, i, j, k, h
  - Request stream: l, l, m, n, o, m
Clairvoyant (Belady): Offline Optimal Caching

• What is the best a caching algorithm could do?
• Offline: uses knowledge of the future
  • (Can’t use in practice)

• Evict the object with the furthest next access time
  • Worst object to keep in the cache

• Three-item cache example:
  • Request stream: h, h, i, j, k, h
  • Request stream: l, l, m, n, o, m

Edge Cache with Different Sizes

Edge Cache with Different Sizes

- "Infinite" size ratio needs 45x of capacity
Edge Cache with Different Algos

- LRU > LFU > FIFO

Edge Cache with Different Algorithms

- S4LRU is a more complex algorithm, uses recency and frequency
Edge Cache with Different Algos

- Clairvoyant (Bélády) shows we can do much better!

Cache Consistency
Some Web Content is Not Cacheable

- **Dynamic content**
  - E.g., stock prices, scores, web cams
- **Content generated by scripts**
  - Results depend on the specific parameters
  - E.g., https://www.google.com/search?q=php+script+url
- **Personalized content**
  - E.g., based on cookie sent by the browser
- **Encrypted content**
  - Cannot decrypt without the appropriate key

Cache Consistency Challenges

- **Web cache needs to know**
  - Whether to cache an item
  - How long to cache an item
  - Whether to check an item’s freshness
  - Whether it is okay to return a stale item
  - Whether the item has sensitive data

- **Server knows the content**
  - Whether the item is dynamic
  - How often the item changes
  - Whether the item has changed
  - Whether stale information is useful
  - Whether item contains sensitive data

**Scalability challenge:** the server cannot remember every client that has cached an item
HTTP Response Header for Cache Control

- Whether to cache
  - no store: no cache should store it
- Who should cache
  - private: only a private cache (e.g., browser)
  - public: any cache, including shared ones
- How long to cache
  - max-age=N: for N seconds
  - must-revalidate: check with the server (don’t return stale item)

**Cache-Control: public, max-age=604800, must-revalidate**

Cache Validation: Client Checks Freshness

- GET /index.html
  - “if <this version> is stale”
- 304 Not Modified

**How do they identify the “version”?**

- Timestamp
  - When the item was modified by the server
  - E.g., Last-Modified: Wed, 21 Oct 2015 07:28:00 GMT
- Version number
  - Entity tag provided by the server
  - E.g., ETag: "33a64df551425fcc55e4d42a148795d9f25f89d4"
Cache Placement

Client Machine (e.g., Browser)

**Advantages**
- Very low latency
- Preserves access bandwidth
- Available when disconnected

**Disadvantages**
- Low hit rate due to “cold” misses
- Many cache consistency checks
- Incomplete logs at the server
Client Network (Forward Proxy Cache)

**Advantages**
- Low latency
- Preserves enterprise bandwidth
- Hits for locally popular content

**Disadvantages**
- Cost to deploy the cache
- Many consistency checks
- Incomplete logs at the server

Server Network (Reverse Proxy Cache)

**Advantages**
- High hit rate across global users
- Greater cooperation with server
- Complete request logs for server
- Preserves server bandwidth

**Disadvantages**
- Costs to deploy the cache
- Does not reduce latency much
- Consumes wide-area bandwidth
Content Distribution Network (CDN)

- Outsourced caching infrastructure
  - Caching for clients and servers
  - Dedicated equipment and software
  - Trained staff, best practices, etc.
- Coordination with the server
  - Generating non-cacheable content
  - Providing detailed measurement data
- Smart cache placement
  - Many caches: handle large request load
  - Close to many clients: reduce latency

CDN Challenges

- Where to place edge sites?
  - Close to many clients, with reasonable cost
- Where to replicate a server’s content?
  - Many edge sites $\rightarrow$ duplicated data
  - Few edge sites $\rightarrow$ larger client latency
- How to direct a client to an edge site?
  - Proximity: for low latency
  - Light load: to reduce congestion
- How to manage each cache?
  - Maximize hit rate?
  - Minimize miss penalty?
  - Fairness across origin servers?
CDN Effectiveness


Conclusions

• Downloading a Web page
  • Name resolution, transport connection, secure session, web messages

• Benefits of caching
  • Reduces user latency, server load, and network bandwidth

• Cache replacement
  • Maximize hit rate by trying to predict the future

• Cache consistency
  • Efficient ways to avoid returning unnecessarily stale responses

• Content distribution networks
  • Caching close to clients, while working on behalf of the servers