Distributed Systems for Content Delivery

Mike Freedman
Lecture 21
COS 418: Distributed Systems

Problem Space

• Many clients accessing web content

• Approach #1: Scale-out web architectures
  – Use many independent instances of stateless web servers
  – Scale-out storage backends via sharding

• Approach #2: Replicate and cache data closer to users
  – Much web content is immutable and/or can be slightly stale

Modern Web Architectures

Stateful Storage Service
Greater fault tolerance
Replication Dimension

Sharding Dimension
Greater scale

System capacity = \( \sum \) individual capacities

Modern Web Architectures

Stateless Routing Layer
Soft-state Caching Layer
Stateful Storage Service
Modern Web Architectures

- Stateless Routing Layer
- Soft-state Caching Layer
- Stateful Storage Service

Types of State

- Soft state – State (information/data) which is used for efficiency, but is not essential for proper operation
  - Soft state can often be regenerated or replaced if needed
  - E.g., data caching is an example of soft state used for performance improvement: If lost, cached data can be refetched from slower, more durable storage

- Hard State – State which is necessary for correctness
  - To date, most of our discussions in class have focused on hard state


Sharded vs. Non-Sharded Caching

- Pros for sharding
  - Greater cache capacity (∑ individual capacities)
  - Adding servers increases both cache capacity and query throughput (although non-sharded can also scale query throughput)

- Cons for sharding
  - Clients need to maintain semi-accurate cache mappings, rather than just random / round-robin selection
  - Elasticity (adding/removing nodes) more complex, either requiring active moving content or cache misses during passive rebalancing
How much to cache?
Many Internet workloads have Power law (Zipf) distribution

Experiences with CoralCDN: A Five-Year Operational View. Freedman. NSDI 2010

Significant benefits at beginning, but then reduced benefit for cache hit rate as cache size grows

Modern Web Architectures
Stateless Routing Layer Soft-state Caching Layer Stateful Storage Service
Caches can also be sharded

Content Delivery Networks
Stateless Routing Layer Soft-state Caching Layer
Downstream Caching
Content Distribution Network

- Proactive content replication
  - Content provider (e.g., CNN) contracts with a CDN
- CDN replicates the content
  - On many servers spread throughout the Internet
- Updating the replicas
  - Reactive by TTL or updates pushed to replicas when the content changes

Caching is Complicated

- Significant fraction (>50%) of distinct HTTP objects may be uncacheable
  - Dynamic data: Stock prices, scores, web cams
  - CGI scripts: results based on passed parameters
  - Cookies: results may be based on passed data
  - Advertising / analytics: want to measure # hits (hint: use random strings)
- Yet significant fraction of HTTP bytes are cacheable
  - Images, video, CSS pages, etc.
- Goal: Maximize cachability, while limiting staleness of cached objects
Caching is powerful: Modern HTTP Video-on-Demand

- Download “content manifest” from origin server
- List of video segments belonging to video
  - Each segment 1-2 seconds in length
  - Client can know time offset associated with each
  - Standard naming for video resolutions/formats: eg, 320dpi, 720dpi, 1040dpi
- Client downloads video segment (at certain resolution) using standard HTTP request.
  - HTTP request can be satisfied by cache: it’s a static object
- Client observes download time vs. segment duration, increases/decreases resolution if appropriate

CDN Case Study: How Akamai Works

Akamai Network
- Servers: ~365,000
- Networks: 1,350
- Countries: 135

https://www.akamai.com/company/facts-figures
How Akamai Uses DNS

cnn.com (content provider)  DNS TLD server

1 2 3 4
ALIAS: g.akamai.net

End user

DNS lookup cache.cnn.com

Akamai global DNS server

Akamai regional DNS server

Nearby Akamai cluster

GET /foo.jpg
Host: cache.cnn.com
How Akamai Uses DNS

- cnn.com (content provider)
- DNS TLD server
- Akamai global DNS server
- Akamai regional DNS server
- Nearby Akamai cluster

GET /foo.jpg
Host: cache.cnn.com

Routing Client Requests within Map

- Mapping system collects data about each “group” of IP addresses, based on network latency, loss, connectivity
  - Updated roughly every minute
    - Short, 60-sec DNS TTLs in Akamai regional DNS accomplish this
  - Map each IP group to a preferred server cluster
  - Map client request to a server in the cluster
    - Load balancer selects a specific server
    - E.g., to maximize the cache hit rate
Selecting server inside cluster

- **Consistent hashing**
  - content_key = hash(URL) mod N
  - node_key = hash(server ID) mod N
  - Content belongs to server’s node_key is “closest” to URL’s content_key

```
CK80
CK20
N105
Server 105
N32
N60
CK80
```

“Consistency”? 
(and/or limiting the staleness of cached objects)

How long should the client cache for?

- Clients (and proxies) cache documents
  - When should origin be checked for changes?
  - Every time? Every session? Date?

- HTTP includes caching information in headers
  - HTTP 0.9/1.0 used: “Expires: <date>”; “Pragma: no-cache”
  - HTTP/1.1 has “Cache-Control”
    - “No-Cache”, “Max-age: <seconds>”
    - “ETag: <opaque value>”

Why the changes between 1.0 and 1.1?

- **Timestamps**
  - Server hints when an object “Expires” (Expires: xxx)
  - Server provides last modified date, client can check if still valid

- **Problems**
  - Client and server might not have synchronized clocks
  - Server replicas might not have synchronized clocks
  - Max-age solves this: relative seconds, not absolute time
What if cache expires?

- Store past expiry time (if room in cache)
- Upon request, first revalidate with server

```
GET / HTTP/1.1
Accept-Language: en-us
If-Modified-Since: Mon, 29 Jan 2001 17:54:18 GMT
Host: www.example.com
Connection: Keep-Alive
```

HTTP/1.1 304 Not Modified
Date: Tue, 27 Mar 2001 03:50:51 GMT
Connection: Keep-Alive

Conditional GETs

- Revalidate cache content if still valid
- Redownload new version if modified

```
GET / HTTP/1.1
Accept-Language: en-us
If-Modified-Since: Mon, 29 Jan 2001 17:54:18 GMT
Host: www.example.com
Connection: Keep-Alive
```

HTTP/1.1 304 Not Modified
Date: Tue, 27 Mar 2001 03:50:51 GMT
Connection: Keep-Alive

Another clock sync problem!

- What if server replicas don’t have aligned modification times?

```
GET / HTTP/1.1
Accept-Language: en-us
If-None-Match: "686897696a7c876b7e"
Host: www.example.com
Connection: Keep-Alive
```

HTTP/1.1 200
Date: Tue, 27 Mar 2001 03:50:51 GMT
ETag: 686897696a7c876b7e

Conclusion

- Content distribution is hard
  - Many, diverse, changing objects
  - Clients distributed all over the world
- Moving content towards client is key
  - Reduces latency, improves throughput, reliability
  - CDNs evolved into complex distributed systems
- Cache controls and revalidation are a key part of managing content freshness with decentralized caching