Web Caching

- Example of the web to illustrate caching and replication issues
  - Simpler model: clients are read-only, only server updates data

Web Proxy Caching

- The principle of cooperative caching.
Consistency Issues

- Web pages tend to be updated over time
  - Some objects are static, others are dynamic
  - Different update frequencies (few minutes to few weeks)
- How can a proxy cache maintain consistency of cached data?
  - Send invalidate or update
  - Push versus pull

Push-based Approach

- Server tracks all proxies that have requested objects
- If a web page is modified, notify each proxy
- Notification types
  - Indicate object has changed [invalidate]
  - Send new version of object [update]
- How to decide between invalidate and updates?
  - Pros and cons?
  - One approach: send updates for more frequent objects, invalidate for rest
Push-based Approaches

• Advantages
  – Provide tight consistency [minimal stale data]
  – Proxies can be passive

• Disadvantages
  – Need to maintain state at the server
    • Recall that HTTP is stateless
    • Need mechanisms beyond HTTP
  – State may need to be maintained indefinitely
    • Not resilient to server crashes

Pull-based Approaches

• Proxy is entirely responsible for maintaining consistency
• Proxy periodically polls the server to see if object has changed
  – Use if-modified-since HTTP messages
• Key question: when should a proxy poll?
  – Server-assigned Time-to-Live (TTL) values
    • No guarantee if the object will change in the interim
Pull-based Approach: Intelligent Polling

• Proxy can dynamically determine the refresh interval
  – Compute based on past observations
    • Start with a conservative refresh interval
    • Increase interval if object has not changed between two successive polls
    • Decrease interval if object is updated between two polls
    • Adaptive: No prior knowledge of object characteristics needed

Pull-based Approach

• Advantages
  – Implementation using HTTP (If-modified-Since)
  – Server remains stateless
  – Resilient to both server and proxy failures

• Disadvantages
  – Weaker consistency guarantees (objects can change between two polls and proxy will contain stale data until next poll)
    • Strong consistency only if poll before every HTTP response
  – More sophisticated proxies required
  – High message overhead
A Hybrid Approach: Leases

- Lease: duration of time for which server agrees to notify proxy of modification
- Issue lease on first request, send notification until expiry
  - Need to renew lease upon expiry
- Smooth tradeoff between state and messages exchanged
  - Zero duration => polling, Infinite leases => server-push
- Efficiency depends on the \( lease \) \textit{duration}.

Policies for Leases Duration

- Age-based lease
  - Based on bi-modal nature of object lifetimes
  - Larger the expected lifetime longer the lease
- Renewal-frequency based
  - Based on skewed popularity
  - Proxy at which objects is popular gets longer lease
- Server load based
  - Based on adaptively controlling the state space
  - Shorter leases during heavy load
Cooperative Caching

- Caching infrastructure can have multiple web proxies
  - Proxies can be arranged in a hierarchy or other structures
    - Overlay network of proxies: content distribution network
  - Proxies can cooperate with one another
    - Answer client requests
    - Propagate server notifications

Hierarchical Proxy Caching

Examples: Squid, Harvest
Locating and Accessing Data

Properties
• Lookup is local
• Hit at most 2 hops
• Miss at most 2 hops (1 extra on wrong hint)

Edge Computing
• Web caches effective when deployed close to clients
  – At the “Edge” of the network
• Edge Computing: paradigm where applications run on servers located at the edge of the network
• Benefits
  – Significantly lower latency than “remote” cloud servers
  – Higher bandwidth
  – Can tolerate network or cloud failures

• Complements cloud computing
  – Cloud providers offer edge servers as well as cloud servers
Edge Computing Origins

• Origins come from mobile computing and web caching
• Content delivery networks
  – Network of edge caches deployed as commercial service
  – Cache web content (especially rich content: images, video)
  – Deliver from closest edge proxy server

• Mobile computing
  – devices has limited resources, limited battery power
  – computational offload: offload work to more capable edge server
  – low latency offload important for interactive mobile applications
  – edge server sends results to the mobile

Content Delivery Networks

• Global network of edge proxies to deliver web content
  – edge clusters of varying sizes deployed in all parts of the world
  – Akamai CDN: 120K servers in 1100 networks, 80 countries
• Content providers are customers of CDN service
  – Examples: news sites, image-rich online stores, streaming sites
  – Decide what content to cache/offload to CDN
  – Embed links to cdn content: http://cdn.com/company/foo.mp4
  – Consistency responsibility of content providers
• Users access website normally
  – Some content fetched by browser from CDN cache
CDN Request Routing

- Web request need to be directed to nearby CDN server
- Two level load balancing
  - Global: decide which cluster to use to serve request
  - Local: decide which server in the cluster to use
- DNS-based approach is common
  - Special DNS server: resolve www.cnn.com/newsvideo.mp4
  - DNS checks location of client and resolves to IP address of nearby CDN server
  - Different users will get resolved to different edge locations

CDN Issues

- Which proxy answers a client request?
  - Ideally the “closest” proxy
  - Akamai uses a DNS-based approach

- Propagating notifications
  - Can use multicast or application level multicast to reduce overheads (in push-based approaches)

- Active area of research
  - Numerous research papers available
CDN Request Processing

- The principal working of the Akamai CDN.

CDN Hosting of Web Applications
Mobile Edge Computing

• Use case: Mobile offload of compute-intensive tasks
• Example: augmented reality, virtual reality (mobile AR/VR)
  – mobile phone or headset has limited resources, limited battery
  – Low latency / response times for interactive use experience
  – mobile devices may lack a GPU or mobile GPU may be limited

• Today’s smartphones are highly capable (multiple cores, mobile GPU, neural processor)
  – mobile offload more suitable for less capable devices (e.g., AR headset)
• 5G cellular providers: deploy edge servers near cell towers
  – industrial automation, autonomous vehicles

Edge Computing Today

• Emerging approach for latency-sensitive applications
• Edge AI: run AI (deep learning) inference at edge
  – home security camera sends feed, run object detection
• Low latency offload: autonomous vehicles, smart city sensors, smart home etc.
• Edge computing as an extension to cloud
  – Cloud regions augmented by local regions
    • Local regions are edge clusters that offer normal cloud service (but at lower latency) E.g., AWS Boston region
  – Internet of Things (IoT) data processing services
Specialized Edge Computing

- Edge accelerators: special hardware to accelerate edge tasks on resource constrained edge servers
  - Nvidia Jetson GPU, Google edge Tensor processing Unit (TUP), Intel Vision Processing Unit (VPU)
- Example: IoT ML inference on edge accelerators
  - Efficient inference on resource-constrained edge servers

Cloud and Edge Architectures

- Offload to cloud, edge, specialized edge,

Traditional cloud (2-tier)

Traditional edge (3-tier)

Specialized (3-tier)