15-441 15-641 Computer Networking

Lecture 16: Delivering Content:
Peer to Peer and CDNs
Peter Steenkiste

Fall 2016
www.cs.cmu.edu/~prs/15-441-F16

Overview

- Web
- Consistent hashing
- Peer-to-peer
  - Motivation
  - Architectures
  - Discussion
- CDN
- Video

2

Distributing Load across Servers

- Given document XYZ, we need to choose a server to use
  - E.g., in a data center
- Suppose we use simple hashing: modulo n of a hash of the name of the document
- Number servers from 1…n
  - Place document XYZ on server (XYZ mod n)
  - What happens when a servers fails? n → n-1
    - Same if different people have different measures of n
  - Why might this be bad?

3

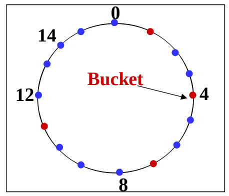
Consistent Hash: Goals

- “view” = subset of all hash buckets that are candidate locations
  - Correspond to a real server
- Desired features
  - Load – all hash buckets have a similar number of objects assigned to them
  - Smoothness – little impact on hash bucket contents when buckets are added/removed
  - Spread – small set of hash buckets that may hold an object regardless of views

4

Consistent Hash – Example

- Construction
  - Assign each of C hash buckets to random points on mod 2^n circle, where, hash key size = n.
  - Map object to random position on unit interval
  - Hash of object = closest bucket



- Monotone → addition of bucket does not cause movement between existing buckets
- Spread & Load → small set of buckets that lie near object
- Balance → no bucket is responsible for large number of objects

5

Consistent Hashing: Ring

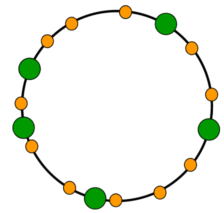
- Use consistent has to map both keys and nodes to an m-bit identifier in the same (metric) identifier space
  - For example, use SHA-1 hashes
  - **Node identifier:** SHA-1 hash of IP address

    IP="198.10.10.1" $\xrightarrow{\text{SHA-1}}$ ID=123

  - **Key identifier:** SHA-1 hash of key

    Key="LetItBe" $\xrightarrow{\text{SHA-1}}$ ID=60

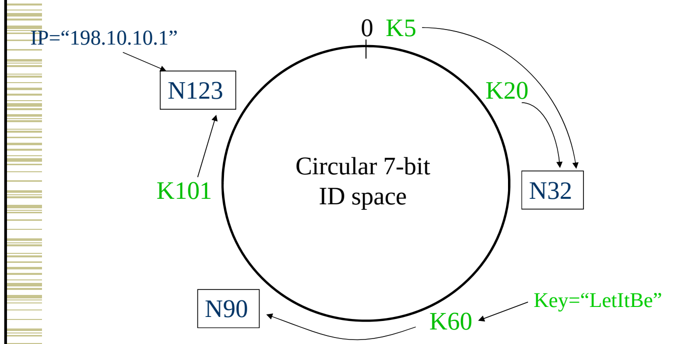
- Also need "rule" for assigning keys to nodes
  - For example: "closest", higher, lower, ..



6

Consistent Hashing Example

Rule: A key is stored at its successor: node with next higher or equal ID



7

Consistent Hashing Properties

- **Load balance:** all nodes receive roughly the same number of keys
  - For N nodes and K keys, with high probability
    - Each node holds at most $(1+\varepsilon)K/N$ keys
    - Provided that K is large compared to N
- When server is added, it receives its initial work load from "neighbors" on the ring
  - "Local" operation: no other servers are affected
  - Similar property when a server is removed

8

Finer Grain Load Balancing

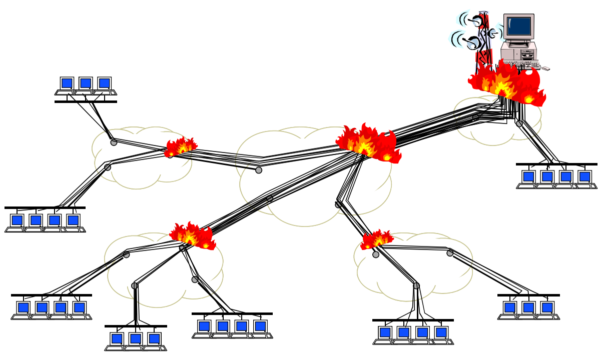
- Redirector knows all server IDs s_i
- It can also track approximate “load” for more precise load balancing
  - Need to define load and be able to track it
- To balance load:
  - W_i = Hash(URL, ip of s_i) for all i
  - Sort W_i from high to low
  - Find first server with low enough load
- Benefits and drawbacks?

Consistent Hashing Used in Many Contexts

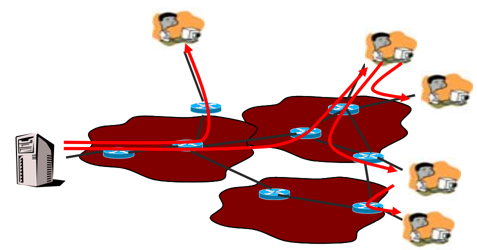
- Distribute load across servers in a data center
  - The redirector sits in data center
- Finding storage cluster for an object in a CDN uses centralized knowledge
  - Why?
  - Can use consistent hashing in the cluster
- Consistent hashing can also be used in a distributed setting
  - P2P systems can use it find files (DHTs)

Scaling Problem

- Millions of clients ⇒ server and network meltdown



P2P System

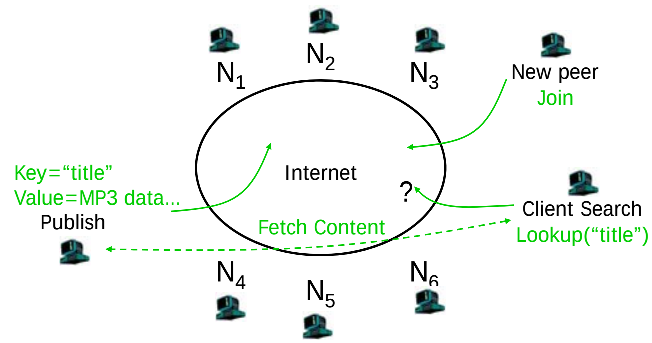


- Leverage the resources of client machines (peers)
  - Computation, storage, bandwidth

Why p2p?

- Harness lots of spare capacity
  - 1 Big Fast Server: 1Gbit/s, $10k/month++
  - 2,000 cable modems: 1Gbit/s, $??
  - 1M end-hosts: Uh, wow.
  - Capacity grows with the number of users!
- Build very large-scale, self-managing systems
  - Same techniques useful for companies and p2p apps
    - E.g., Akamai's 14,000+ nodes, Google's 100,000+ nodes
  - Many differences to consider
    - Servers versus arbitrary nodes
    - Hard state (backups!) versus soft state (caches)
    - Security, fairness, freeloading, ..

Common P2P Framework



What is (was) out there?

	Central	Flood	Super-node flood	Route
Whole File	Napster	Gnutella		Freenet
Chunk Based	BitTorrent		KaZaA	DHTs eDonkey 2000

When are p2p Useful?

- Works well for caching and "soft-state", read-only data
  - Works well! BitTorrent, KaZaA, etc., all use peers as caches for hot data
- Difficult to extend to persistent data
  - Nodes come and go: need to create multiple copies for availability and replicate more as nodes leave
- Not appropriate for search engine styles searches
  - Complex intersection queries ("the" + "who"): billions of hits for each term alone
  - Sophisticated ranking: Must compare many results
- Search time tends to be unpredictable

Overview

- Web
- Consistent hashing
- Peer-to-peer
- CDN
  - Motivation
  - Edge servers
  - Content delivery
  - Mapping
  - Impact on Internet
- Video

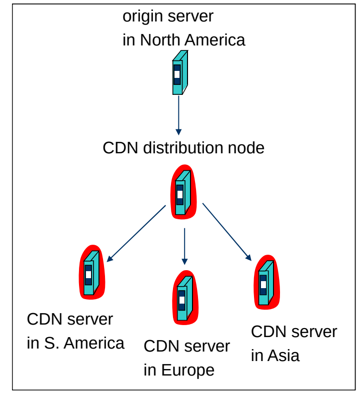
Some slides based on presentation by Patrick Gilmore

Content Distribution Networks (CDNs)

- The content providers are the CDN customers.

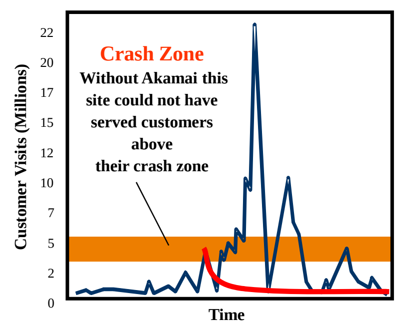
Content replication

- CDN company installs hundreds of CDN servers throughout Internet
  - Close to users
- CDN replicates its customers' content in CDN servers. When provider updates content, CDN updates servers



http://www.akamai.com/html/technology/nui/news/index.html

Case Study on Reliability and Scalability: The 2000 Election



What is the CDN?

- Edge Caches: work with ISP and networks everywhere to install edge caches
  - Edge = close to customers
- Content delivery: getting content to the edge caches
  - Content can be objects, video, or entire web sites
- Mapping: find the “closest” edge server for each user and deliver content from that server
  - Network proximity not the same as geographic proximity
  - Focus is on performance as observed by user (quality)

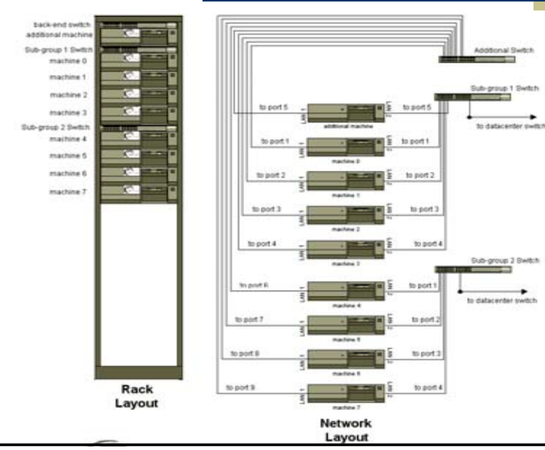
Potential Benefits

- Very good scalability
  - Near infinite if deployed properly
- Good economies at large scales
  - Infrastructure is shared efficiently by customers
  - Statistical multiplexing: hot sites use more resources
- Can reduce latency – more predictable performance
  - Through mapping to closest server
  - Avoids congestion and long latencies
- Can be extremely reliable
  - Very high degree of redundancy
  - Can mitigate some DoS attacks

Edge Caches

- Region – set of caches managed as a cluster
  - May have a specific function: http, streaming, …
- Availability is a major concern in architecture design
- Redundancy at the network level
  - See next slide
- Dealing with server failures
  - Servers do fail occasionally
  - Each server has a “buddy” which is constantly trading hellos
  - If hellos stop, buddy starts to respond directly to requests for primary server
  - Users in the middle of a download may have to hit “reload”
  - No one else will notice any interruption

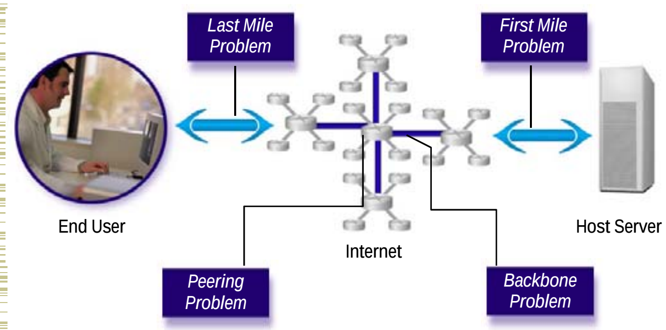
Example Configuration



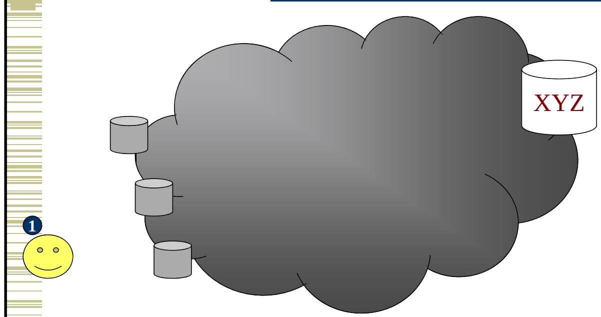
Overview

- Web
- Consistent hashing
- Peer-to-peer
- CDN
  - Motivation
  - Edge servers
  - Content delivery
  - Mapping
  - Impact on Internet
- Video

Content Delivery: Possible Bottlenecks

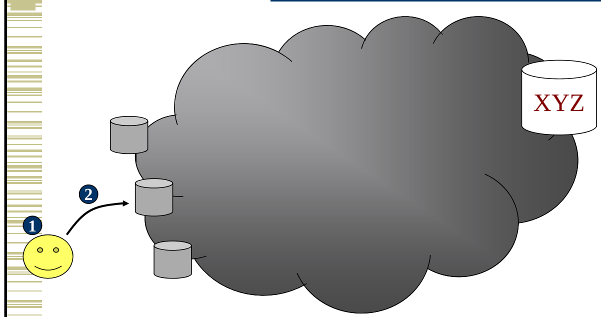


Process Flow



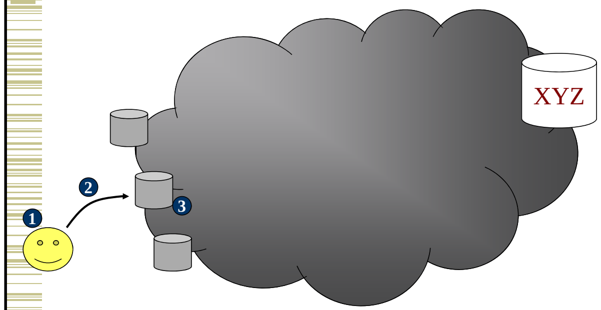
1. User wants to download distributed web content

Process Flow



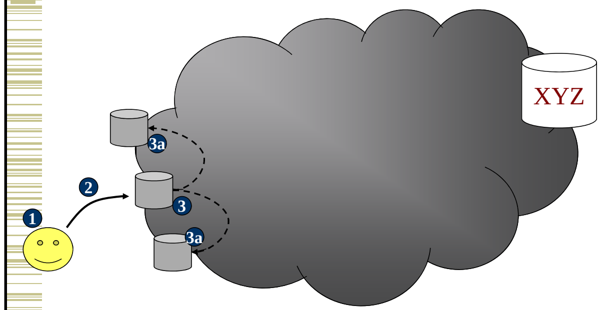
2. User is directed through Akamai’s dynamic mapping to the “closest” edge cache

Process Flow



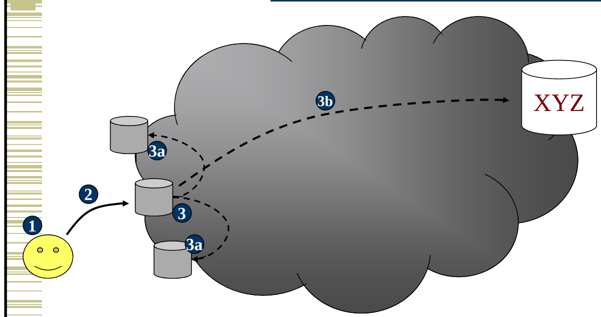
3. Edge cache searches local hard drive for content

Process Flow



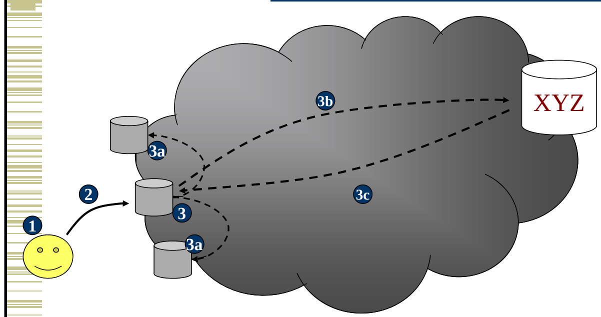
3b. If requested object is not on local hard drive, edge cache checks other edge caches in same region for object

Process Flow



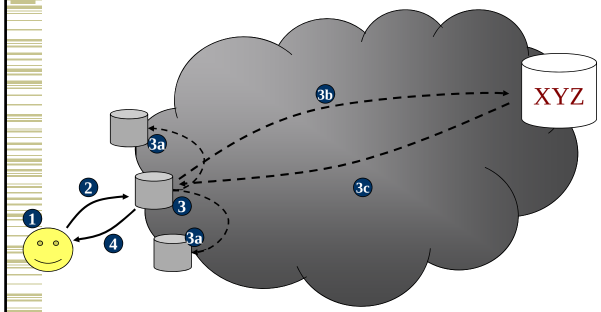
3b. If requested object is not cached or not fresh, edge cache sends an HTTP GET the origin server

Process Flow



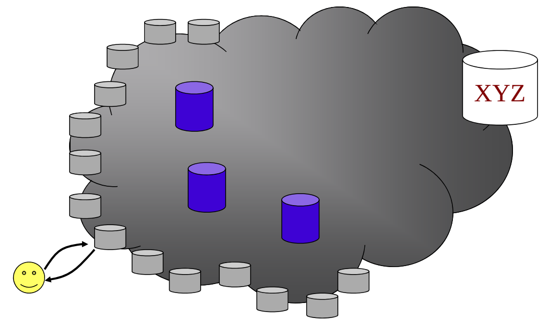
3c. Origin server delivers object to edge cache over optimized connection

Process Flow



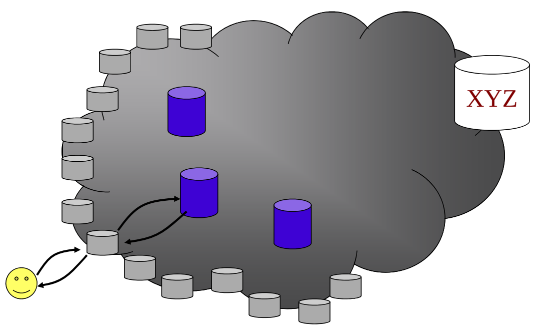
4. Edge server delivers content to end user

Core Hierarchy Regions



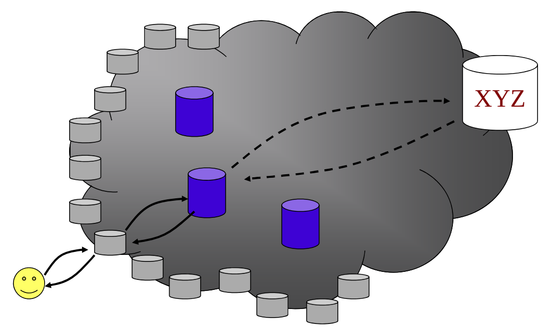
1. User requests content and is mapped to optimal edge Akamai server

Core Hierarchy Regions



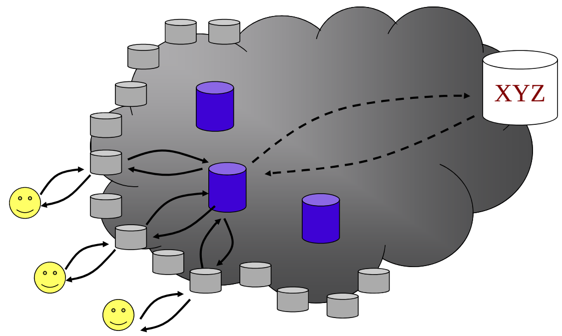
2. If content is not present in the region, it is requested from most optimal core region

Core Hierarchy Regions



3. Core region makes one request back to origin server

Core Hierarchy Regions



4. Core region can serve many edge regions with one request to origin server

Core Hierarchy Features

Reduces traffic back to origin server

- Reduces infrastructure needs of customer
- Provides best protection against flash crowds
  - Especially important for large files (*e.g.* Operating System updates or video files)

Improved end-user response time

- Core regions are well connected
- Optimized connection speeds object delivery

Overview

- Web
- Consistent hashing
- Peer-to-peer
- CDN
  - Motivation
  - Edge servers
  - Content delivery
  - Mapping
  - Impact on Internet
- Video

39