

CDNs & Replication

Prof. Vern Paxson
EE122 Fall 2007
TAs: Lisa Fowler, Daniel Killebrew,
Jorge Ortiz

Improving HTTP Performance:
Caching w/ Content Distribution Networks

- Integrate forward and reverse caching functionality
 - One overlay network (usually) administered by one entity
 - e.g., Akamai
- Provide document caching
 - **Pull**: Direct result of clients' requests
 - **Push**: Expectation of high access rate
- Also do some processing
 - Handle *dynamic* web pages
 - *Transcoding*

Improving HTTP Performance:
Caching with CDNs (cont.)

Improving HTTP Performance:
CDN Example – Akamai

Akamai:

- Creates new domain names for each client content provider
 - e.g., `i.a.cnn.net = CNAME custom.i.cnn.net.edgesuite.net`
`custom.i.cnn.net.edgesuite.net = CNAME a1921.g.akamai.net`

Customer/Content Provider:

- Modifies content
 - Embedded URLs reference new domains
 - CNN page's HREF's refer to `i.a.cnn.net`
- Pushes content out to Akamai as it changes
 - Or: Akamai pulls it on demand w/ usual caching mech.
 - Both CNN & Akamai have control over load distribution

Improving HTTP Performance:
CDN Example – Akamai

"Akamaized" response object has inline URLs for secondary content at (after resolving CNAMEs) `a1921.g.akamai.net` and other Akamai-managed DNS names.

akamai.net DNS servers
Akamai servers store/cache secondary content for "Akamaized" services.

lookup `a1921.g.akamai.net`

GET `http://cnn.com`
1 - DNS Lookup
2 - Fetch page w/ "Akamaized" content
3 - DNS Lookup for Akamai URLs
4 - Fetch content

Improving HTTP Performance:
Caching vs. Replication

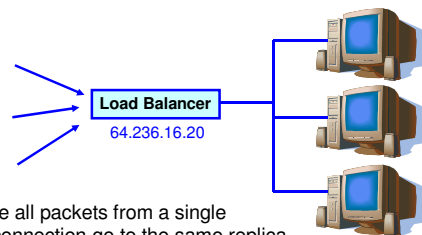
- Why move content closer to users?
 - Reduce latency for the user
 - Reduce load on the network and the server
- How?
 - Caching
 - Replicate content "on demand" *after* a request
 - Store the response message locally for future use
 - Challenges:
 - May need to verify if the response has changed
 - ... and some responses are not cacheable
 - Replication
 - Planned* replication of content in multiple locations
 - Update of resources handled *outside* of HTTP
 - Can replicate scripts that create dynamic responses

Hosting: Multiple Machines Per Site

- Replicate a popular Web site across multiple machines
 - Helps to handle the load
 - Places content closer to clients
 - Helps when content *isn't cacheable* by proxies/CDNs
- Problem: Want to direct client to a *particular* replica
 - Why?
 - Balance load across server replicas
 - Pair clients with *nearby* servers
- Solution #1: Manual selection by clients
 - Each replica has its own site name
 - A Web page lists the replicas (e.g., by name, location)
 - ... and asks clients to click on a hyperlink to pick

Hosting: Multiple Machines Per Site

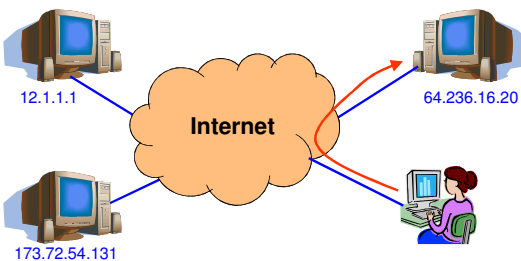
- Solution #2: single IP address, multiple machines
 - Run multiple machines behind a single IP address



- Ensure all packets from a single TCP connection go to the same replica

Hosting: Multiple Machines Per Site

- Solution #3: multiple addresses, multiple machines
 - Same name but different addresses for all of the replicas
 - Configure DNS server to return different addresses



Hosting: Multiple Sites Per Machine

- Multiple Web sites on a single machine
 - Hosting company runs the Web server on behalf of multiple sites (e.g., `www.foo.com` and `www.bar.com`)
- Problem: `GET /index.html`
 - `www.foo.com/index.html` or `www.bar.com/index.html`?
- Solutions:
 - Multiple server processes on the same machine
 - Have a separate IP address (or port) for each server
 - Include site name in HTTP request
 - Single Web server process with a single IP address
 - Client includes "Host" header (e.g., `Host: www.foo.com`)
 - Required header with HTTP/1.1