Last Time

- RPC invoke semantics
  - At least once
  - At most once

Understanding Your Workload

- Engineering principle
  - Make the common case fast, and rare cases correct
  - (From Patterson & Hennessy books)
  - This principle cuts through generations of systems.
- Example?
  - CPU Cache
- Knowing common cases == understanding your workload
  - E.g., read dominated? Write dominated? Mixed?

Content Distribution Problem

- Power law (Zipf distribution)
  - Models a lot of natural phenomena
  - Social graphs, media popularity, wealth distribution, etc.
  - Happens in the Web too.

Content Distribution Workload

- What are the most frequent things you do on Facebook?
  - Read/write wall posts/comments/likes
  - View/upload photos
  - Very different in their characteristics
- Read/write wall posts/comments/likes
  - Mix of reads and writes so more care is necessary in terms of consistency
  - But small in size so probably less performance sensitive
- Photos
  - Write-once, read-many so less care is necessary in terms of consistency
  - But large in size so more performance sensitive

Facebook’s Photo Distribution Problem

- “Hot” vs. “very warm” vs. “warm” photos
  - Hot: Popular, a lot of views
  - Very warm: Somewhat popular, still a lot of views
  - Warm: Unpopular, but still a lot of views in aggregate
“Hot” Photos
- How would you serve these photos?
  - Caching should work well.
    - Many views for popular photos
  - Where should you cache?
    - Close to users
  - What’s commonly used these days?
    - CDN
      - CDN mostly relies on DNS, so we’ll look at DNS then CDN.
  - (Very warm and warm: next two lectures)

CSE 486/586 Administrivia
- Nothing much!

Domain Name System (DNS)
Proposed in 1983 by Paul Mockapetris

Separating Names and IP Addresses
- Names are easier (for us!) to remember
  - www.cnn.com vs. 64.236.16.20
- IP addresses can change underneath
  - Move www.cnn.com to 173.15.201.39
  - E.g., renumbering when changing providers
- Name could map to multiple IP addresses
  - www.cnn.com to multiple replicas of the Web site
- Map to different addresses in different places
  - Address of a nearby copy of the Web site
  - E.g., to reduce latency, or return different content
- Multiple names for the same address
  - E.g., aliases like ee.mit.edu and cs.mit.edu

Two Kinds of Identifiers
- Host name (e.g., www.cnn.com)
  - Mnemonic name appreciated by humans
  - Provides little (if any) information about location
  - Hierarchical, variable # of alpha-numeric characters
- IP address (e.g., 64.236.16.20)
  - Numerical address appreciated by routers
  - Related to host’s current location in the topology
  - Hierarchical name space of 32 bits

Hierarchical Assignment Processes
- Host name: www.cse.buffalo.edu
  - Domain: registrar for each top-level domain (e.g., .edu)
  - Host name: local administrator assigns to each host
- IP addresses: 128.205.32.58
  - Prefixes: ICANN, regional Internet registries, and ISPs
  - Hosts: static configuration, or dynamic using DHCP
Overview: Domain Name System

• A client-server architecture
  – The server-side is still distributed for scalability.
  – But the servers are still a hierarchy of clients and servers
• Computer science concepts underlying DNS
  – Indirection: names in place of addresses
  – Hierarchy: in names, addresses, and servers
  – Caching: of mappings from names to/from addresses
• DNS software components
  – DNS resolvers
  – DNS servers
• DNS queries
  – Iterative queries
  – Recursive queries
• DNS caching based on time-to-live (TTL)

Strawman Solution #1: Local File

• Original name to address mapping
  – Flat namespace
  – /etc/hosts
  – SRI kept main copy
  – Downloaded regularly
• Count of hosts was increasing: moving from a machine per domain to machine per user
  – Many more downloads
  – Many more updates

Strawman Solution #2: Central Server

• Central server
  – One place where all mappings are stored
  – All queries go to the central server
• Many practical problems
  – Single point of failure
  – High traffic volume
  – Distant centralized database
  – Single point of update
  – Does not scale

Need a distributed, hierarchical collection of servers

Domain Name System (DNS)

• Properties of DNS
  – Hierarchical name space divided into zones
  – Distributed over a collection of DNS servers
• Hierarchy of DNS servers
  – Root servers
  – Top-level domain (TLD) servers
  – Authoritative DNS servers
• Performing the translations
  – Local DNS servers
  – Resolver software

DNS Root Servers

• 13 root servers (see http://www.root-servers.org/)
• Labeled A through M
  – A Verisign, Dulles, VA
  – C Cogent, Herndon, VA (also Los Angeles)
  – D USDoD Vienna, VA
  – E NASA M.Vale, CA
  – F Internet Software, Palo Alto, CA (and 17 other locations)
  – G US DoD Vienna, VA
  – H ARL Aberdeen, MD
  – I Autonomica, Stockholm (plus 3 other locations)
  – J Verisign, Dulles, VA (also Los Angeles)
  – K RIPE London (+ Amsterdam, Frankfurt)
  – L ICANN Los Angeles, CA
  – M INWIDE Tokyo

TLD and Authoritative DNS Servers

• Top-level domain (TLD) servers
  – Generic domains (e.g., com, org, edu)
  – Country domains (e.g., uk, fr, ca, jp)
  – Typically managed professionally
  – Network Solutions maintains servers for “com”
  – Educause maintains servers for “edu”
• Authoritative DNS servers
  – Provide public records for hosts at an organization
  – For the organization’s servers (e.g., Web and mail)
  – Can be maintained locally or by a service provider
DNS Caching

- Performing all these queries take time
  - And all this before the actual communication takes place
  - E.g., 1-second latency before starting Web download
- Caching can substantially reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.cnn.com) visited often
  - Local DNS server often has the information cached
- How DNS caching works
  - DNS servers cache responses to queries
  - Responses include a "time to live" (TTL) field
  - Server deletes the cached entry after TTL expires

Using DNS

- Local DNS server ("default name server")
  - Usually near the end hosts who use it
  - Local hosts configured with local server (e.g., /etc/resolv.conf) or learn the server via DHCP
- Client application
  - Extract server name (e.g., from the URL)
  - Do gethostbyname() to trigger resolver code
- Server application
  - Extract client IP address from socket
  - Optional gethostbyaddr() to translate into name

Example

Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

Recursive vs. Iterative Queries

- Recursive query
  - Ask server to get answer for you
  - E.g., request 1 and response 8
- Iterative query
  - Ask server who to ask next
  - E.g., all other request-response pairs

Negative Caching

- Remember things that don't work
  - Misspellings like www.cnn.comm and www.cnnn.com
  - These can take a long time to fail the first time
  - Good to remember that they don't work
  - ... so the failure takes less time the next time around
DNS Resource Records

**DNS:** distributed db storing resource records (RR)

- **Type=A**
  - name is hostname
  - value is IP address
- **Type=NS**
  - name is domain
  - value is hostname of authoritative name server for this domain
- **Type=CNAME**
  - name is alias for some “canonical” (real) name:
    - Example: www.ibm.com is really srveast.backup2.ibm.com
  - value is canonical name
- **Type=MX**
  - value is name of mailserver associated with name

Inserting Resource Records into DNS

- **Example:** just created startup “FooBar”
- Register foo.bar.com at Network Solutions
  - Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com TLD server:
    - (foo.bar.com, dns1.foo.bar.com, NS)
    - (foo.bar.com, dns1.foobar.com, A)
- Put in authoritative server dns1.foobar.com
  - Type A record for www.foo.bar.com
  - Type MX record for foo.bar.com
- Play with “dig” on UNIX

Reliability

- **DNS servers are replicated**
  - Name service available if at least one replica is up
  - Queries can be load balanced between replicas
- **UDP used for queries**
  - Need reliability: must implement this on top of UDP
- **Try alternate servers on timeout**
  - Exponential backoff when retrying same server
- **Same identifier for all queries**
  - Don’t care which server responds

```bash
$ dig nytimes.com ANY
;; QUESTION SECTION:
:nytimes.com. 192.31.80.30

;; SERVER: 192.52.178.30#53(192.52.178.30)
Query time: 103 msec
;; AUTHORITY SECTION:
:nytimes.com. 2009070102 1800 3600 604800 3600
~all"
```
dig nytimes.com ANY +norec @ns1t.nytimes.com

; <<>>HEADER<<- opcode: QUERY, status: NOERROR, id: 39107
;; flags: qr aa; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
nytimes.com.

;; ANSWER SECTION:
nytimes.com.      300           IN  A    199.239.137.245
nytimes.com.      300           IN  A    199.239.136.200
nytimes.com.      300           IN  A    199.239.136.245
nytimes.com.      300           IN  TXT  "v=spf1 mx ptr ip4:199.239.138.0/24
include:alerts.wallst.com include:authsmtp.com ~all"

;; ADDITIONAL SECTION:
ns1t.nytimes.com.   300           IN  A    199.239.137.15

Query time: 10 msec
SERVER: 199.239.137.15#53(199.239.137.15)
WHEN: Mon Feb 23 11:25:20 2009
MSG SIZE  rcvd: 454

Content Distribution Networks (CDNs)

• Content providers are CDN customers

Content replication
• CDN company installs thousands of servers throughout Internet
  – In large datacenters
  – Or, close to users
• CDN replicates customers’ content
• When provider updates content, CDN updates servers

Server Selection

• Which server?
  – Lowest load: to balance load on servers
  – Best performance: to improve client performance
  – Based on what? Location? RTT? Throughput? Load?
  – Any alive node: to provide fault tolerance
• How to direct clients to a particular server?
  – As part of routing: anycast, cluster load balancer
  – As part of application: HTTP redirect
  – As part of naming: DNS

How Akamai Works

End-user

GET index.html

HTTP

Inn.com (content provider)

DNS root server

Akamai global DNS server

Akamai regional DNS server

Cache.inn.com

Akamai cluster

End-user

How Akamai Works

End-user

DNS lookup

Inn.com (content provider)

DNS root server

Akamai global DNS server

Akamai regional DNS server

Inn.com

Akamai cluster
How Akamai Works

1. End-user requests content from cnn.com.
2. DNS lookup is performed by the user's DNS server.
3. The request is directed to Akamai's global DNS server.
4. The request is then directed to Akamai's regional DNS server.
5. The request is finally directed to an Akamai cluster.
6. The content is delivered to the end-user.

Summary

- DNS as an example client-server architecture
- Why?
  - Names are easier (for us!) to remember
  - IP addresses can change underneath
  - Name could map to multiple IP addresses
  - Map to different addresses in different places
  - Multiple names for the same address
- Properties of DNS
  - Distributed over a collection of DNS servers
- Hierarchy of DNS servers
  - Root servers, top-level domain (TLD) servers, authoritative DNS servers
Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC), Michael Freedman (Princeton), and Jennifer Rexford (Princeton).