CSE 454

HTTP + Server Architecture

Previously

- Introduction
- History
- Networking
  - IP
  - TCP
  - DNS

Outline

- HTTP Protocol
- Service Architecture & Scaling

For next time
- Mailing list
- Reading
  - HTTP Made easy
  - Responsibilities
  - Mercator

Connecting on the WWW

What happens when you click?

- Suppose
  - You are at www.yahoo.com/index.html
  - You click on www.grippy.org/mattmarg/
- Browser uses DNS => IP addr for www.grippy.org
- Opens TCP connection to that address
- Sends HTTP request:

  Request
  Get /mattmarg/ HTTP/1.0
  User-Agent: Mozilla/2.0 (Macintosh; i; PPC)
  Accept: text/html; */*
  Cookie: name = value
  Referer: http://www.yahoo.com/index.html
  Host: www.grippy.org
  Expires: 
  If-modified-since: 

  Headers

HTTP Response

HTTP/1.0 200 Found
Date: Mon, 10 Feb 1997 23:48:22 GMT
Server: Apache/1.1.1 HotWired/1.0
Content-type: text/html
Last-Modified: Tues, 11 Feb 1999 22:45:55 GMT

- One click => several responses
  - HTTP1.0: new TCP connection for each elt/page
  - HTTP1.1: KeepAlive - several requests/connection
Response Status Lines

- **1xx** Informational
- **2xx** Success
  - 200 Ok
- **3xx** Redirection
  - 302 Moved Temporarily
- **4xx** Client Error
  - 404 Not Found
- **5xx** Server Error

HTTP Methods

- **GET**
  - Bring back a page
- **HEAD**
  - Like GET but just return headers
- **POST**
  - Used to send data to server to be processed (e.g., CGI)
  - Different from GET:
    - A block of data is sent with the request, in the body, usually with extra headers like Content-Type and Content-Length:
    - Request URL is not a resource to retrieve; it's a program to handle the data being sent
    - HTTP response is normally program output, not a static file.
  - PUT, DELETE, ...

Cookies

- Small piece of info
  - Sent by server as part of response header
  - Stored on disk by browser; returned in request header
  - May have expiration date (deleted from disk)
- Associated with a specific domain & directory
  - Only given to site where originally made
  - Many sites have multiple cookies
  - Some have multiple cookies per page!
- Most Data stored as name=value pairs
- See
  - C:\Program Files\Netscape\Users\default\cookies.txt
  - C:\WINDOWS\Cookies

Logging Web Activity

- Most servers support “common logfile format” or “extended logfile format”
- Apache lets you customize format
- Every HTTP event is recorded
  - Page requested
  - Remote host
  - Browser type
  - Referring page
  - Time of day
- Applications of data-mining logfiles ??

Connecting on the WWW

![Diagram showing the connection between Web Browser, Web Server, and Internet]

Client-Side View

- Content rendering engine
  - Tags, positioning, movement
- Scripting language interpreter
  - Document object model
  - Events
- Programming language itself
- Link to custom Java VM
- Security access mechanisms
- Plugin architecture + plugins
Server-Side View
- Database-driven content
- Lots of Users
- Scalability
- Load balancing
- Often implemented with cluster of PCs
- 24x7 Reliability
- Transparent upgrades

Trade-offs in Client/Server Arch.
- Compute on clients?
  - Complexity: Many different browsers
    - (Firefox, IE, Safari, …) \times Version \times OS
  - Compute on servers?
    - Peak load, reliability, capital investment.
    - Access anywhere, anytime, any device
    - Groupware support (shared calendar, …)
    - Lower overall cost (utilization & debugging)
    - Simpler to update service

Dynamic Content
- We want to do more via an http request
  - E.g., we’d like to invoke code to run on the server.
- Initial solution: Common Gateway Interface (CGI) programs.
- Example: web page contains form that needs to be processed on server.

CGI Code
- CGI scripts can be in any language.
- A new process is started (and terminated) with each script invocation (overhead!).
- Improvement I:
  - Run some code on the client’s machine
  - E.g., catch missing fields in the form.
- Improvement II:
  - Server APIs (but these are server-specific).

Java Servlets
- Servlets: applets that run on the server.
  - Java VM stays, servlets run as threads.
- Accept data from client + perform computation
- Platform-independent alternative to CGI.
- Can handle multiple requests concurrently
  - Synchronize requests - use for online conferencing
- Can forward requests to other servers
  - Use for load balancing

Java Server Pages (JSP) Active Server Pages (ASP)
- Allows mixing static HTML w/ dynamically generated content.
- JSP is more convenient than servlets for the above purpose.
**Tiered Architectures**

1-tier = dumb terminal → smart server.
2-tier = client/server.
3-tier = client/application server/database.

Why decompose the server?

---

**Two-Tier Architecture**

TIER 1: CLIENT

TIER 2: SERVER

Server performs all processing

Client workstation

Web Server

Application Server

Database Server

Database

Server does too much work. Weak Modularity.

---

**Three-Tier Architecture**

TIER 1: CLIENT

TIER 2: SERVER

TIER 3: BACKEND

Application server offloads processing to Tier 3

Web Server + Application Server

Database Server

Database

Using 2 computers instead of 1 can result in a huge increase in simultaneous clients. Depends on % of CPU time spent on database access. While DB server waits on DB, Web server is busy!

---

**Getting to 'Giant Scale’**

- Only real option is cluster computing

Optional Backplane:
System-wide network for intra-server traffic:
Query redirect, coherence traffic for store, updates, ...

---

**Cluster Computing**

<table>
<thead>
<tr>
<th>Service</th>
<th>Nodes</th>
<th>Queries</th>
<th>Node Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOL Web Cache</td>
<td>&gt;1000</td>
<td>10B/day</td>
<td>4 CPU DEC 4100s</td>
</tr>
<tr>
<td>Inktomi Search Eng</td>
<td>&gt;1000</td>
<td>80M/day</td>
<td>2 CPU Sun wkstns</td>
</tr>
<tr>
<td>Geocities</td>
<td>&gt;300</td>
<td>25M/day</td>
<td>PC-based</td>
</tr>
<tr>
<td>Web email</td>
<td>&gt;5000</td>
<td>1B/day</td>
<td>Free BSD PCs</td>
</tr>
</tbody>
</table>

---

**Assumptions**

- Service provider has limited control
  - Over clients, network
- Queries drive system
  - HTTP Get
  - FTP
  - RPC
- Read Mostly
  - Even at Amazon, browsing >> purchases

---

From: Brewer Lessons from Giant-Scale Services
**Cluster Computing: Benefits**

- **Absolute Scalability**
  - Large % of earth population may use service!

- **Incremental Scalability**
  - Can add / replace nodes as needed
  - Nodes ~5x faster / 3 year depreciation time
  - Cap ex $$ vs. cost of rack space / air cond

- **Cost & Performance**
  - But no alternative for scale: hardware cost << ops

- **Independent Components**
  - Independent faults help reliability

---

**Load Management**

- **Round-Robin DNS**
  - Problem: doesn’t hide failed nodes

- **Layer 4 switch**
  - Understand TCP, port numbers

- **Layer 7 (application layer) switch**
  - Understand HTTP; Parse URLs at wire speed!
  - Use in pairs (automatic failover)

- **Custom front-ends**
  - Service-specific layer 7 routers in software

- **Smart client end-to-end**
  - Hard for WWW in general. Used in DNS, Cell roaming

---

**Case Studies**

Simple Web Farm  
Inktomi (2001) Supports programs (not users)  
Persistent data is partitioned across servers:  
⇑ capacity, but ⇓ data loss if server fails

---

**High Availability**

- **Essential Objective**
  - Phone network, railways, water system

- **Challenges**
  - Component failures
  - Constantly evolving features
  - Unpredictable growth

---

**Typical Cluster**

- Extreme symmetry
- Internal disks
- No monitors
- No visible cables
- No people!
- Offsite management

---

**Availability Metrics**

- **Traditionally: Uptime**
  - Uptime = (MTBF – MTTR)/MTBF

- **Phone system ~ “Four or Five Nines”**
  - Four nines means 99.99% reliability
  - I.e. less than 60 sec downtime / week

- **How improve uptime?**
  - Measuring "MTBF = 1 week" requires > 1 week
  - Measuring MTTR much easier
  - New features reduce MTBF, but not MTTR
  - Focus on MTTR, just best effort on MTBF
Yield
• Queries completed / queries offered
  - Numerically similar to uptime, but
  - Better match to user experience
  - (Peak times are much more important)

Harvest
• Data available / complete data
  - Fraction of services available
    - E.g. Percentage of index queried for Google
    - Ebay seller profiles down, but rest of site ok

Architecture
• What do faults impact? Yield? Harvest?
• Replicated systems
  Faults → reduced capacity (hence, yield @ high util
• Partitioned systems
  Faults → reduced harvest
  Capacity (queries / sec) unchanged

DQ Principle
exists physical bottleneck
Data/Query × Queries/Sec = Constant

Using DQ Values
• Measurable, Tunable
• Absolute Value Irrelevant
  - Relative value / changes = predictable!
• Methodology
  1. Define DQ value for service
  2. Target workload & load generator
  3. Measure for hardware × software × DB size
     Linearity: small cluster (4 nodes) predict perf for 100
  4. Plan: capacity/traffic; faults: replic/part;

Graceful Degradation
• Too expensive to avoid saturation
• Peak/average ratio
  - 1.6x - 6x or more
  - Ticketmaster: 10x capacity for Phantom Menace
    • Not enough...
  • Dependent faults (temperature, power)
    • Overall DQ drops way down
  • Cutting harvest by 2 doubles capacity...

Admission Control (AC) Techniques
• Cost-Based AC
  - Denying an expensive query allows 2 cheap ones
    - Inktomi
• Priority-Based (Value-Based) AC
  - Stock trades vs. quotes
    - Datek
• Reduced Data Freshness

Managing Evolution
• Traditional Wisdom
  - "High availability = minimal change"
• Internet: continuous growth, features
  - Imperfect software (memory leaks, intermit bugs
• Acceptable quality
  - Target MTBF; low MTTR; no cascading failures
  - Maintenance & upgrades = controlled failures

[Graph showing DQ value changes over time]