Database-Powered Web Servers

Alexandros Labrinidis
Univ. of Pittsburgh

labrinid@cs.pitt.edu
Examples of db-powered web sites

- Google.com (search engine)
- Amazon.com (shopping)
- eBay.com (auctions)
- WellsFargo.com (online banking)
- weather.com (forecasts)
- expedia.com (travel)
- NSF.gov (proposal submission)
- my.yahoo.com (personalized newspaper)
- NYTimes.com (electronic newspaper)
Why is WebDB so popular?

- Arguments from web “producers”:
  - easy to “publish” databases over the Web
  - wealth of information available
  - enable personalization
  - allow targeted advertising

- Arguments from web “consumers”:
  - no need to install special software
  - easy to learn - uniform user interface
  - personalized content

- Today – even seemingly static sites have WebDB
Typical WebDB architecture – 3 tiers

- **Web server**: handle HTTP requests
- **Application server**: web workflow
- **DB server**: data storage & queries
Typical WebDB architecture – 2 tiers

2-tiers: incorporate application server within web server
HTTP: HyperText Transfer Protocol

Client request:

nitrọgen{4} telnet www.google.com 80
Trying 216.239.37.101...
Connected to www.google.com.
Escape character is '^]'.
GET /index.html HTTP/1.0
[two carriage returns]
HTTP: HyperText Transfer Protocol

Server response:

HTTP/1.0 200 OK
Content-Length: 2532
Connection: Close
Server: GWS/2.0
Date: Thu, 07 Nov 2002 16:57:59 GMT
Content-Type: text/html
Cache-control: private

Set-Cookie:
    PREF=ID=24bce47555c1db8b:TM=1036688279:LM=1036688279:S=t4XqRr3VPTPWKMEp; expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.google.com

<html>
    ....
</html>

Connection closed by foreign host.
HTML: HyperText Markup Language

- page wrapped in `<html></html>`

- many formatting commands:
  - `<font color="red">SOMETHING IN RED</font>`
  - CSS: Cascading StyleSheets

- Form example:

```html
<form action="/activate.cgi" method=POST>
Please give your name:
<input type=text name="username"
    size=15 maxlength=30>
<input type=submit value="Submit Name">
</form>
```
**CGI: Common Gateway Interface**

- Introduced in the early 90s
- Protocol for exchanging *form data* between client and server

User requests form → Web Server sends form to user

User submits form → CGI forwards form data to application

User receives output → Web Server sends output to user
CGI encoding

- **method=GET**
  - Form data are encoded as part of the URL
    (e.g. http://www.google.com/search?q=cmu)

- **method=POST**
  - Form data are passed via environment variables

- **encoding of query string:**
  - (name,value) pairs from FORM
  - name1=value1&name2=value2&...
  - space → +
  - other controls characters in hexadecimal format
Cookies

- HTTP is connection-less
  - a new connection is established with every request
  - HTTP/1.1 supports persistent connections (but not popular)

Q: How to maintain state over a session?
A: Cookies

- Cookies are text-only strings that are stored at the browsers’ memory (and disk)

- .google.com  TRUE / FALSE 2147368447  PREF
  ID=0a2612f3162aa05f:TM=1035999053:LM=1035999053:S=ko0-i9cOsU6nMaW6
Cookies and Databases

- Use cookie to store client-ID

- Storing client-ID enables personalization
  - Example: NYTimes.com

- Storing client-ID enables access to restricted areas
  - Example: NYTimes.com (subscription-based access to articles)

- Use cookie to prohibit duplicate submissions on polls
  - Randomly generated ID
  - Matched against database of those already “voted”
Cookies and Privacy

- Cookies only store text-strings given by servers
  - Include domain, expiration date, etc

- Only a server from the same domain can access a previously stored cookie

- The case of Doubleclick:
  - Same cookie used by an advertising agency
  - Match individuals with browsing profiles that span multiple sites
  - Huge data mining opportunity
  - Huge controversy
Client-side programs

- Extend web user interface by running programs at clients
  - Allow for sophisticated UIs
  - Must be careful of malicious code (from untrusted servers)

- Java
  - Full-fledged programming language
  - Protection capabilities

- Other client-side scripting languages:
  - JavaScript (SUN)
  - VBScript (MSFT)
  - Flash/Shockwave (Macromedia)
Server-side programs

- Implement server applications and workflow
  - Programs that generate HTML
  - Embedded HTML

- Java:
  - Servlets
  - Java Server Pages (JSP – SUN)

- Server-side scripting:
  - Active Server Pages (ASP – MSFT)
  - PHP
  - mod_perl
Embedded HTML: PHP

- PHP stands for “PHP Hypertext Processor”

Example:

```html
<html>
<head> <title>Example</title> </head>
<body>
<?php echo "Hi, I'm a PHP script!"; ?> </body>
</html>
```
Generating HTML: mod_perl

- mod_perl: efficiently use perl to generate HTML
- http://perl.apache.org

Example:

```perl
#!/bin/perl
print << EOF;
<html>
<head> <title>Example</title>
</head>
<body>
Hi, I was generated by mod_perl!
</body>
</html>
EOF
```
DB interface: mod_perl

- Two modules – allow for portability:
  - DBI: database independent library
  - DBD: database dependent driver

- Example:

```perl
use DBI;
my $dbh = DBI->connect("oracle", "user", "pass");
my $stmt = $dbh->prepare("SELECT * from foo;");
$stmt->execute();
while (@row = $stmt->fetchrow_array()) {
    print "Row: @row\n";
}
$dbh->disconnect();
```
Improving Performance: CGI

- CGI forks new process on every call
  - context switch
  - re-connect to db server

- mod_perl: maintain pool of processes
  - no context switch – assign call to existing process
  - scales better – 10 times faster
    [Labrinidis & Roussopoulos, SIGMOD Record, Mar 2000]

- mod_perl: re-use db connections
  - no need to re-connect to db server
  - twice faster than mod_perl
Improving Performance: Caching

- **Web Caching**
  - store static content close to the users
  - avoid re-transmitting over the network
  - consistency: Time-To-Live (TTL)

- **Dynamic Web Caching**
  - store dynamic content (from db) and re-use if possible
  - avoid re-computing from database
  - what to cache:
    - entire web pages
    - query results
    - HTML fragments