CIS 455/555: Internet and Web Systems

Web Servers

September 15, 2021
Plan for today

- Servers with threads
- Consistency issues with shared resources
- Event-driven servers
- Programming abstractions for Web servers
- State, and where to keep it
handleNewConnection(e) { startReading(e.connection); }
handleRequestRead(e) {
    if (e.request == "GET <document>") {
        issueFilesystemRead(document);
    } else {
        issueWrite(e.connection, "HTTP/1.1 400 Bad req");
    }
}
/* other handlers go here */

main() {
    EventQueue q;
    while (true) {
        e = q.getNextEvent();
        case e of {
            NewConnection: handleNewConnection(e);
            RequestArrived: handleRequestRead(e);
            FileReadCompleted: handleFileRead(e);
            AllDataWritten: handleDataWritten(e);
        }
    }
}
What does the server code look like?
Many ways to react to users:
- Javascript
- Forms and the POST method, GET method, etc.
Dynamic content in the backend

- How can we make content dynamic?
  - Web server needs to return different web pages, depending on how the user interacts with the web application

- Idea #1: Build a monolithic piece of code
  - Specialized to every web site and application
  - Pros/cons?

- Idea #2: Spin off a thread/process
  - Common Gateway Interface (CGI)
  - Oracle and MySQL databases
  - Pros/cons?
Handling Web Requests in an Extensible, Modular Fashion

- Web server “listens” on port 80 – “daemon” task

- Upon a request, it needs to invoke a response

- How should that response task get executed?
  - First question: Threads vs events vs hybrid schemes
  - Second question: How do we make the handler a modular thing?
Example “Control Flow” from Homework 1

Web Server

Spark Controller

cmd-line args

Web Service

Server Setup
Example “Control Flow” from Homework 1

Web Server → Spark Controller → Web Service

Server Setup

ServerSocket Connections → HTTP Listener

cmd-line args

HTTP Task w/ Socket → HTTP Task Queue

HTTP Worker

Server thread pool
Example “Control Flow” from Homework 1
The HTTP Request Handler

- “Default” option can be a handler that just matches the URL path with a filesystem path – and returns static content
  - You’ll always need this, for images, static HTML, etc.

- But sometimes we need to request dynamic content, or trigger behavior on the server (e.g., handle a form)
The Spark Approach

- Processing a form (and dynamic pages, and REST calls) requires that a *piece of code* run in response to an HTTP request

- A common design pattern: *routes* and *filters* for the non-default case
  - Think of event-driven programming
  - We receive a request that maps to a particular path or “route”
  - We execute a *route handler* with the request info
Routes in Different Languages

Respond with Hello World! on the homepage:

```javascript
app.get('/', function (req, res) {
  res.send('Hello World!')
})
```

Node.js: JavaScript

Django: Python

```python
router = routers.SimpleRouter()
router.register(r'users', UserViewSet)
router.register(r'accounts', AccountViewSet)
class UserViewSet(ModelViewSet):
  ...
  @detail_route(methods=['post'], permission_classes=[IsAdminOrIsSelf], url_path='change-password')
def set_password(self, request, pk=None):
```

© 2021 A. Haeberlen, Z. Ives, V. Liu

University of Pennsylvania
• We register the handler, a Lambda function

• The route includes parameters (":name") and the request can include queries

• The return value from the function is the HTTP body, but we can also modify the response
Filters in Java
Spark Framework, sparkjava.com

```java
before((request, response) -> {
    boolean authenticated;
    // ... check if authenticated
    if (!authenticated) {
        halt(401, "You are not welcome here");
    }
});
```

- Similar to routes -> register a Lambda function
- Occurs before or after every request
- Commonly used to check for invalid requests or authentication without needing to copy the code everywhere
  - A continuation!
Plan for today

- Programming abstractions for Web servers
- State, and where to keep it
Session state

- What if clients make multiple related requests?
  - Example: Open file, read data, read more data, close file
  - Need to remember some state between requests, e.g., which file was opened, or how much data has already been read
  - Who should keep this state: Client, server, or both?
  - If it is kept on the client, how does the server access it?
  - If it is on the server, how does the client reference it?

- If there is no state, or the client keeps all of it, we can build a stateless server
  - Server can forget everything about completed requests
  - Pros and cons of such a design?
HTTP cookies

- What is a **cookie**?
  - A set of key-value pairs that a web site can store in your browser (example: 'sessionid=12345')
  - Created with a Set-Cookie header in the HTTP response
  - Browser sends the cookie in all subsequent requests to the same web site until it expires
A few more words on cookies

Each cookie can have several attributes:

- An expiration date
  - If not specified, defaults to end of current session
- A domain and a path

Browser only sends the cookies whose path and domain match the requested page

- Why this restriction?

... Set-Cookie: sessionid=12345;
  expires=Thu, 02-Feb-2017 23:59:59 GMT;
  path=/;
  domain=.cis.upenn.edu
...

© 2021 A. Haeberlen, Z. Ives, V. Liu
University of Pennsylvania
Note that cookies are not secure!

- What if someone can listen in on our traffic?
  - Firesheep: Captures WiFi packets and extracts session cookies, e.g., for Facebook and Twitter
  - Can be used to 'hijack' sessions (illegal!!!)
  - Why does this work? How could it be prevented?
What are cookies being used for?

- Many useful things:
  - Convenient session management (compare: URL rewriting)
  - Remembering user preferences on web sites
  - Storing contents of shopping carts etc.

- Some problematic things:
  - Storing sensitive information (e.g., passwords)
  - Tracking users across sessions & across different web sites to gather information about them
Using cookies to keep track of state

- Store the state in client
  - Use the browser’s cookie storage as a key-value store

- Store the state in the webserver
  - E.g., with a Session object

- What about shared, persistent state?
Shared, persistent state

- A database back-end is often a good option
  - Recovery and reliability features
  - Transaction support
  - Simple query interface

- Often the database is on a different server from the executing code
  - This is what Enterprise JavaBeans are designed to support: distributed transactions
  - “Model-view-controller” pattern is the most common
Interfacing with a database

- A very common operation:
  - Read some data from a database, output in a web form
  - e.g., postings on reddit, items for a product catalog, etc.

- Three problems, abstracted away by ODBC/ADO/JDBC:
  - Impedance mismatch from relational DBs to objects in Java
  - Standard API for different databases
  - Physical implementation for each DB
The DoubleClick cookie

To help our partners manage their advertising and websites, we offer many products, including AdSense, AdWords, Google Analytics, and a range of DoubleClick-branded services. When you visit a page or see an ad that uses one of these products, either on Google services or on other sites and apps, various cookies may be sent to your browser.

(Source: http://www.google.com/privacy_ads.html)

- **Used by the Google Display Network**
  - DoubleClick used to be its own company, but was acquired by Google in 2008 (for $3.1 billion)

- **Tracks users across different visited sites**
  - Associates browser with 'relevant interest categories'
The Evercookie

- Arms race:
  - Advertisers want to track users
  - Privacy-conscious users do not want to be tracked

- What if users simply delete cookies?
  - Most browsers offer convenient dialogs and/or plugins
  - But: Cookies are not the only way to store data in browsers
  - Example: ETags

- Proof of concept: The 'evercookie'
  - Stores cookie in eight separate ways: HTTP cookies, Flash cookies, force-cached PNGs, web history (!), HTML5 session storage, HTML5 local storage, HTML5 global storage, HTML5 database storage
  - If any of the eight survives, it recreates the others

Recap: State, and where to keep it

- **Session state**
  - Could be kept on the client, the server, or both
  - HTTP cookies are a common way (but not the only way) to keep session state on the client
  - Cookies are used for many other things, e.g., tracking

- **Persistent state**
  - Usually kept in a database on the server side (locally or on a separate machine)
  - JDBC/ODBC are standard software interfaces for accessing databases independent of programming languages, OSes, etc.
Plan for today

- Programming abstractions for Web servers
- State, and where to keep it
- Virtualization basics
  - Emulation, binary translation
  - Hypervisors and virtual machines
  - Hardware support
- Containers
  - Union filesystems
  - Docker Hub
  - Mesos, Docker Swarm, Kubernetes
Modern servers and the Cloud

- We’ve seen how to build a server application
- But how to host and scale out the server?

- Given:
  - Dependencies on specific OS versions, software environments
  - Hardware failures, load imbalances, ...

- For a more hands-on experience, see:
  https://rominirani.com/docker-tutorial-series-a7e6ff90a023
Scaling services

- “Old” model: Host your own server
- As you scale, buy and install more servers...
Idea: Consolidation

- Could we “pool together” many servers?
  - E.g., at Penn?

An old idea

- Hardware-level virtual computers since ’60s
- IBM S/360, IBM VM/370 mainframe systems
- Timeshare multiple single-user OS instances on expensive hardware
Idea: Build a “PC emulator”

Key challenges

- **Heterogeneity**: Many different apps, OS stacks, etc.
- **Isolation**: Different applications shouldn’t be able to consume each other’s resources
- **Demultiplexing**: How to make sure requests go to the right place
  - Need to be able to run multiple applications concurrently! Would be good to be able to “context-switch”, just like between processes
  - Load will vary! Would be good to have the ability to migrate software from one machine to another
Idea: Build a “PC emulator”

- **Version 1: Interpreter for machine code**
  - Write a program that *emulates* the CPU and devices
  - Reads and interprets each instruction in the machine code for the OS, software, device drivers, etc.
  - E.g., Nintendo/Playstation emulators

- **Version 2: Binary translation**
  - Cross-compile the original machine code to the instruction set of the machine’s actual CPU
  - Can be static (compile-time) or dynamic (runtime)
  - Used in the past (e.g., Apple’s 68K→PowerPC, PowerPC→x86)
  - Today: Android ART, Microsoft’s fledgling x86-64-on-ARM
Virtualization

What if we emulate a machine with the same instruction set?

- No need to cross-compile! But...
- Remaining challenge is to emulate “Ring 0” – privileged mechanisms for memory management, protection, etc.

Originally done with clever software tricks

- VMware etc.

Today: Hardware support available

- Example: Intel VT-x
- Essentially provides a “ring -1” for the “hypervisor” or “VMM”