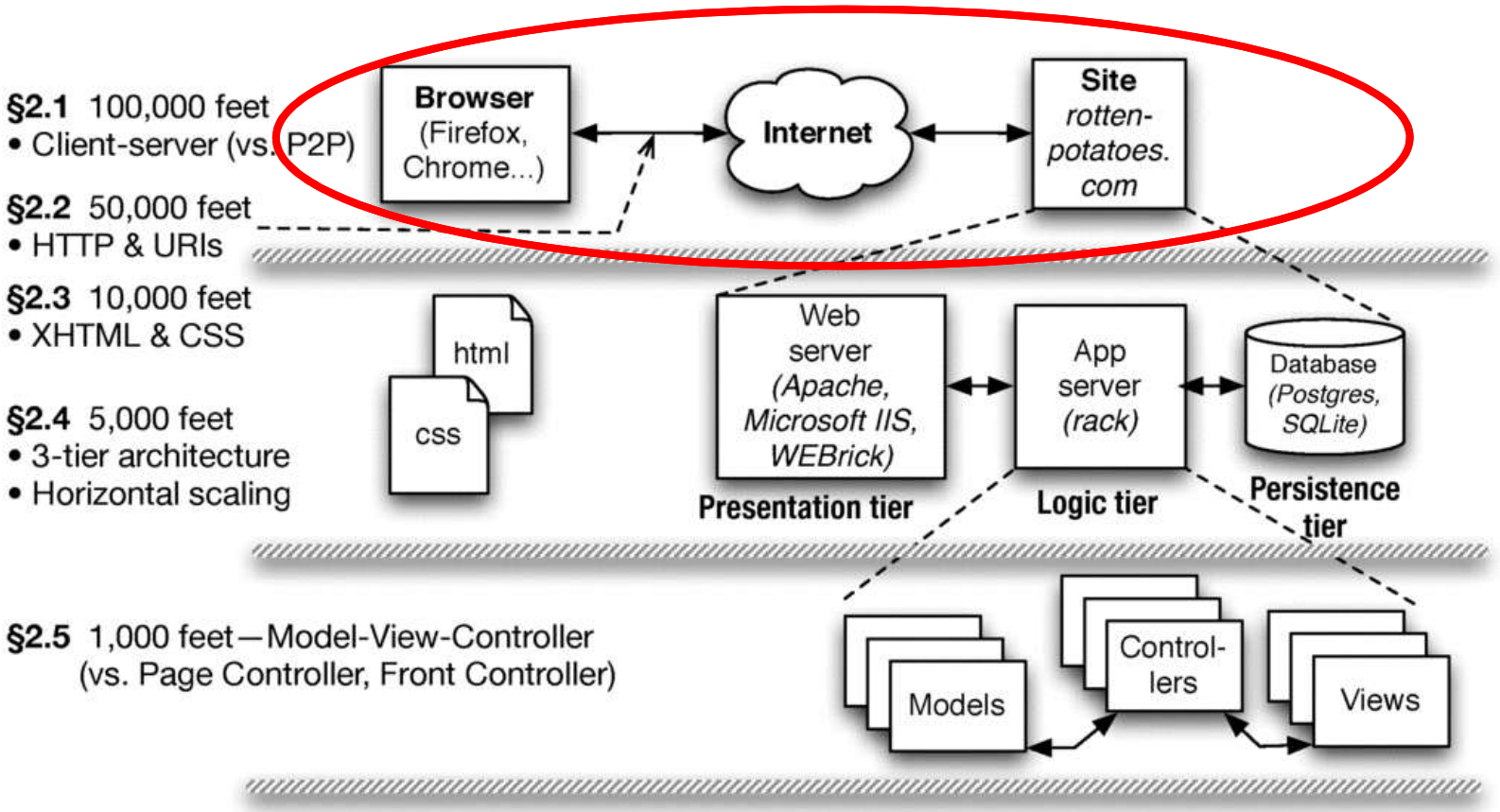
The Architecture of SaaS Applications

Chung-Kil Hur

(Credit: Byung-Gon Chun & Many Slides from UCB CS169 taught by Armando Fox and David Patterson)

SWPP, CSE, SNU

The Web as a Client-Server System; TCP/IP intro

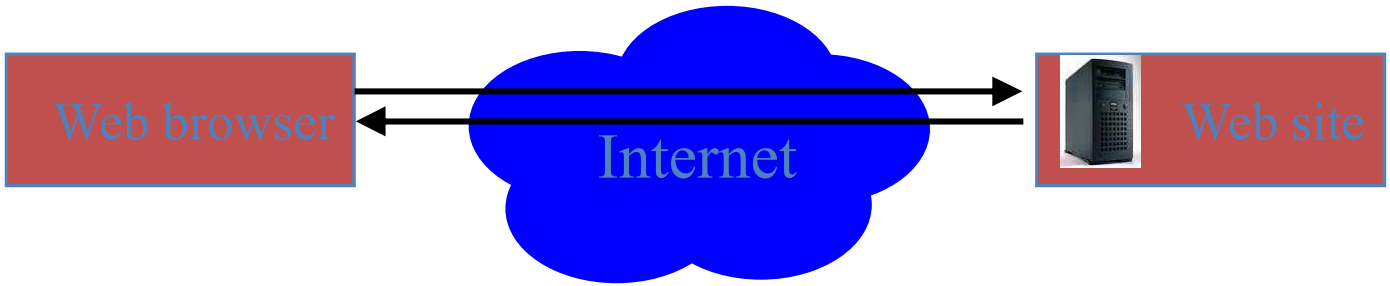


§2.6 500 feet: Active Record models (vs. Data Mapper)
§2.7 500 feet: RESTful controllers (Representational State Transfer for self-contained actions)
§2.8 500 feet: Template View (vs. Transform View)

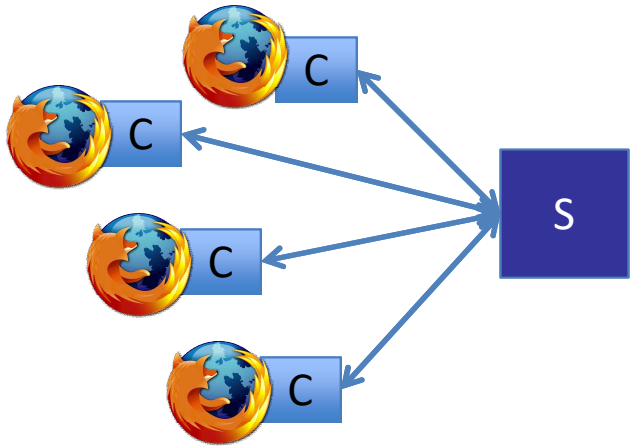
- **Active Record**
- Data Mapper
- **REST**
- **Template View**
- Transform View

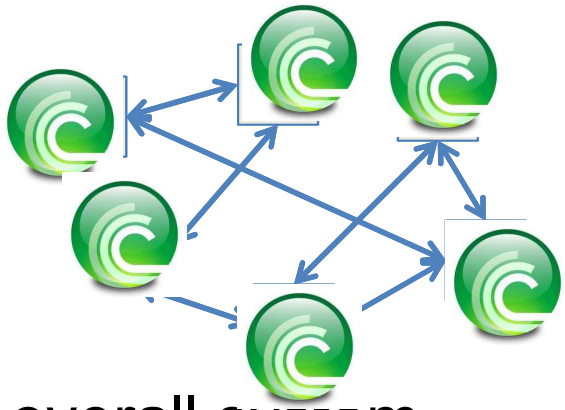
Web at 100,000 feet

- The web is a c*lient/server* architecture
- It is fundamentally *request/reply oriented*



Client-Server vs. Peer-to-Peer

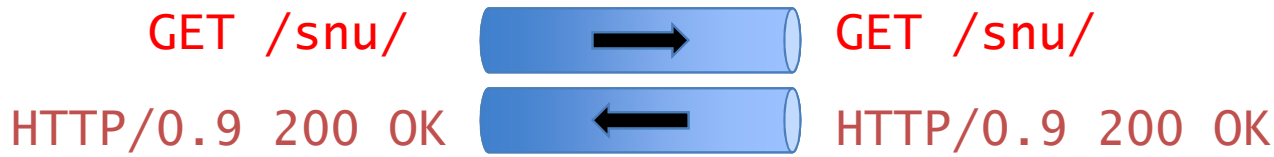




- High-level architecture of the overall system
  - Soon we'll talk about architecture "inside" boxes
- Client & server each *specialized* for their tasks
  - Client: ask questions on behalf of users
  - Server: wait for & respond to questions, serve many clients
- Design Patterns capture common structural solutions to recurring problems
  - Client-Server is an *architectural pattern*

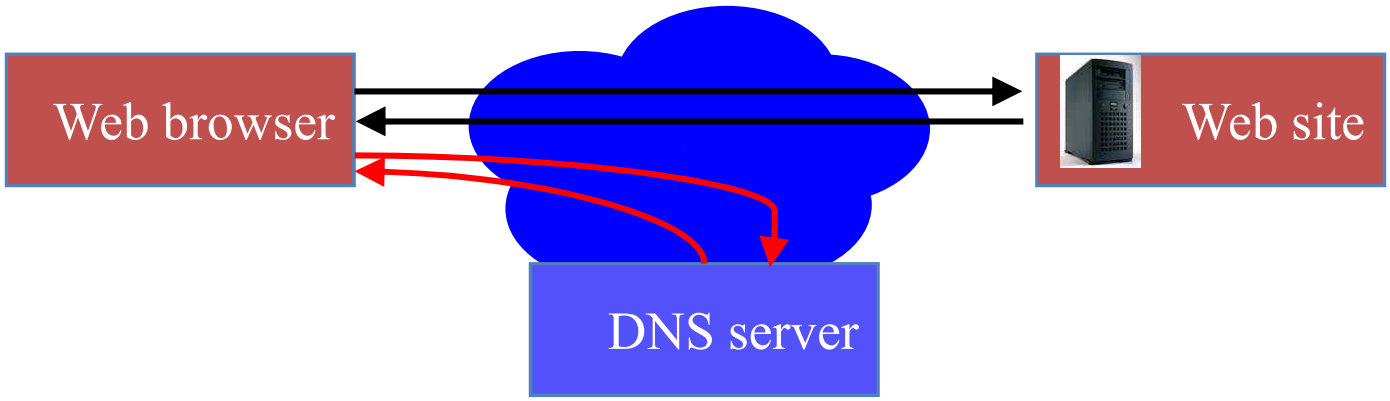
Nuts and bolts: TCP/IP protocols

- IP (Internet Protocol) *address* identifies a physical network interface with four *octets,* e.g. `128.32.244.172`
  - Special address `127.0.0.1` is "this computer", named `localhost`, even if not connected to the Internet!
- TCP/IP (Transmission Control Protocol/Internet Protocol)
  - IP: no-guarantee, best-effort service that delivers *packets* from one IP address to another
  - TCP: make IP reliable by detecting "dropped" packets, data arriving out of order, transmission errors, slow networks, etc., and respond appropriately
  - TCP *ports* allow multiple TCP apps on same computer
- Vint Cerf & Bob Kahn: 2004 Turing Award for Internet architecture & protocols, incl. TCP/IP



Web at 100,000 feet

- The web is a *client/server* architecture
- It is fundamentally *request/reply oriented*
- Domain Name System (DNS) is another kind of server that maps *names* to *IP addresses*



Now that we're talking, what do we say?

Hypertext Transfer Protocol

- an *ASCII-based request/reply protocol* for transferring information on the Web
- *HTTP request* includes:
  - *request method* (`GET`, `POST`, etc.)
  - Uniform Resource Identifier (URI)
  - HTTP protocol version understood by the client
  - *headers*—extra info regarding transfer request
- *HTTP response* from server
  - Protocol version & Status code =>
  - Response headers
  - Response body

HTTP status codes:
2xx — *all is well*
3xx — *resource moved*
4xx — *access problem*
5xx — *server error*

Cookies

- Observation: *HTTP is stateless*
- Early Web 1.0 problem: how to guide a user “through” a flow of pages?
  - use IP address to identify returning user?
    - ✖ public computers, users sharing single IP
  - embed per-user junk into URI query string?
    - ✖ breaks caching
- Quickly superseded by *cookies*

Client Pull vs. Server Push

- WebSockets and HTML5 have some support for allowing the server to push updated content to the client
  - Periodic polling

- True server push – allow the server to initiate connection to the client to wake it up when new information is available

HTML, CSS, JavaScript

- HTML
  - Describes and defines the content of a webpage
- CSS (Cascading Style Sheets)
  - Describes how HTML elements are to be displayed on screen, paper, or in other media.
- JavaScript

JavaScript

- A programming language that is run by most modern browsers
- A high-level, dynamic, weakly typed, object-based, and interpreted programming language
- Alongside HTML and CSS, JavaScript is one of the three core technologies of the web client side
- It can be used to control web pages on the client side of the browser, server-side programs, and even mobile applications.

Progression for the JavaScript Community

- From plain scripts

1. Adding module systems – maintainability, avoiding namespace pollution, reusability
2. Adding compilers (transpilers)
    1. Writing in a language that "thinks" the way you do makes you more productive
    2. Use next generation JavaScript, today
    3. FB Babel, MS TypeScript compiler
3. Adding type systems
    1. MS Typescript - a superset that compiles down to JavaScript — although it feels almost like a new statically-typed language in its own right
    2. FB Flow - an open-source static type checking library, incrementally add types to your JavaScript code

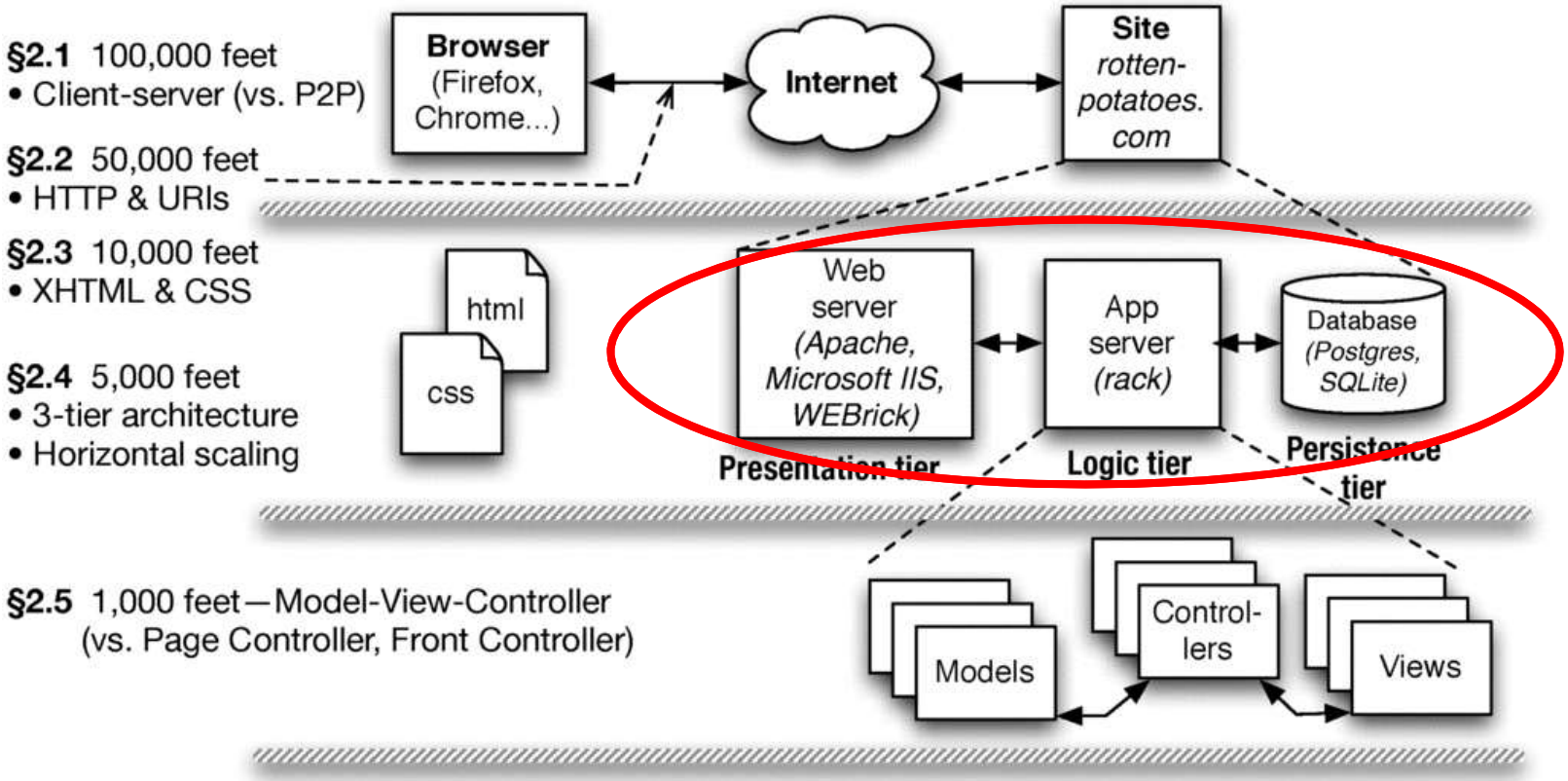
TypeScript

- A super-set of JavaScript
- Add classical object-oriented semantics
- Types
- Classes
- Interfaces
- Inheritance
- Modules
- Generics

Client Application Framework

- Easily implement interactive web applications
- Client-side MVC framework
- Angular: a framework for building client applications in HTML and either JavaScript or a language like TypeScript that compiles to JavaScript.
- React/Redux: Javascript Library for Building User Interfaces

3-tier shared-nothing architecture & scaling



§2.6 500 feet: Active Record models (vs. Data Mapper)
§2.7 500 feet: RESTful controllers (Representational State Transfer for self-contained actions)
§2.8 500 feet: Template View (vs. Transform View)

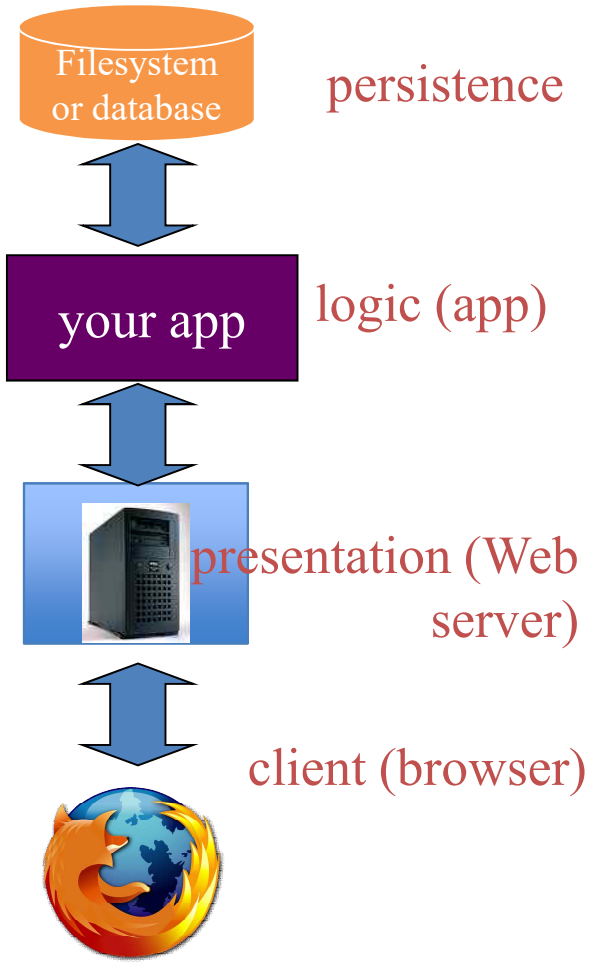
- **Active Record**
- Data Mapper
- **REST**
- **Template View**
- Transform View

Dynamic content generation

- In the Elder Days, most web pages were (collections of) plain old files
- But most interesting Web 1.0/e-commerce sites *run a program* to generate each "page"
- Originally: templates with embedded code "snippets"
- Eventually, code became "tail that wagged the dog" and moved out of the Web server

Sites that are really programs (SaaS)

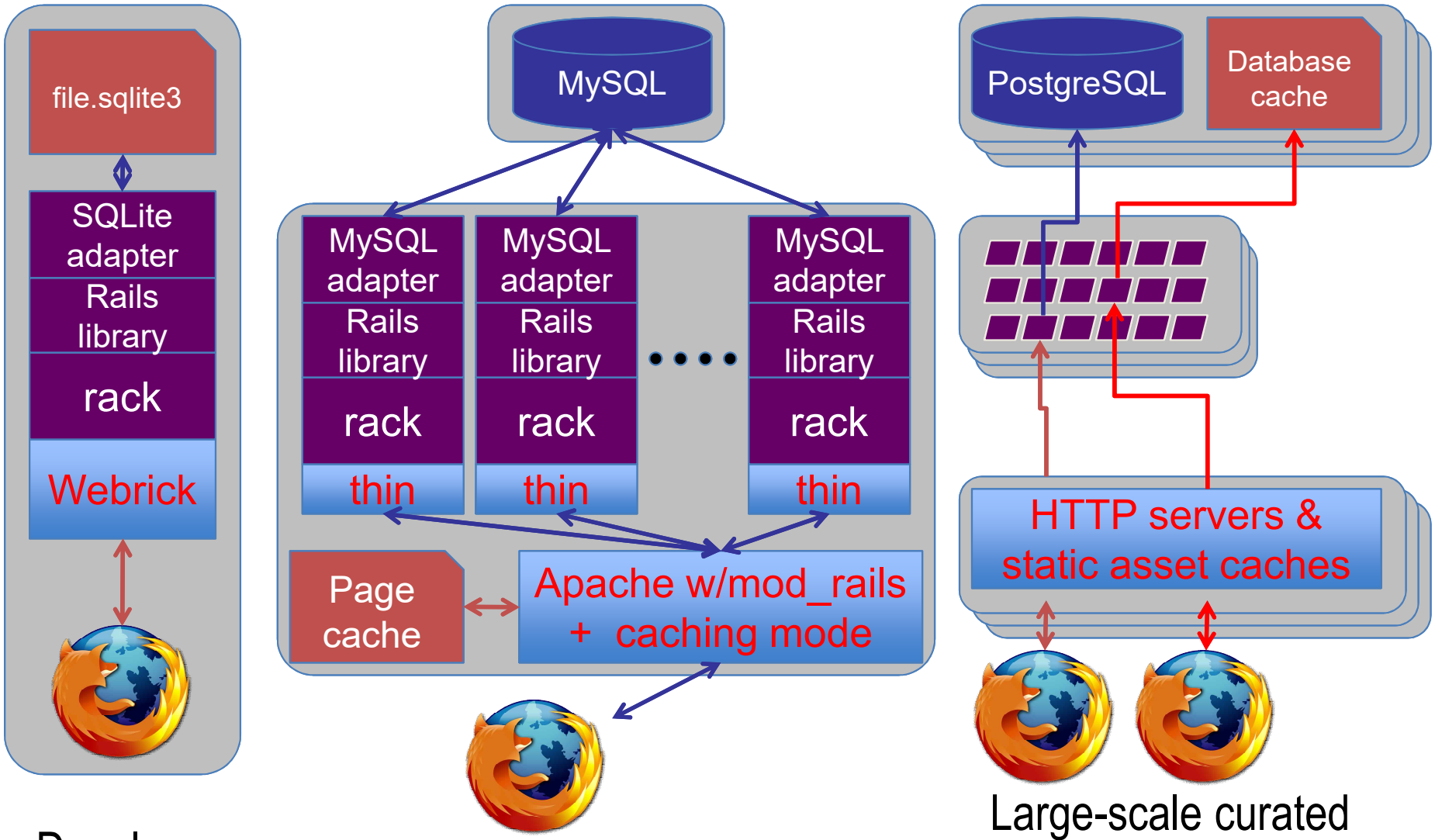
- How do you:
  - "map" URI to correct program & function?
  - pass arguments?
  - invoke program on server?
  - handle persistent storage?
  - handle cookies?
  - handle errors?
  - package output back to user?
- *Frameworks* support these common tasks
  - *Python Django*
  - *Node.js*



Web Application Framework

- URL routing
- HTML, XML, JSON, and other output format templating
- Database manipulation
- Security against Cross-site request forgery (CSRF) and other attacks
- Session storage and retrieval

Developer environment vs. medium-scale deployment

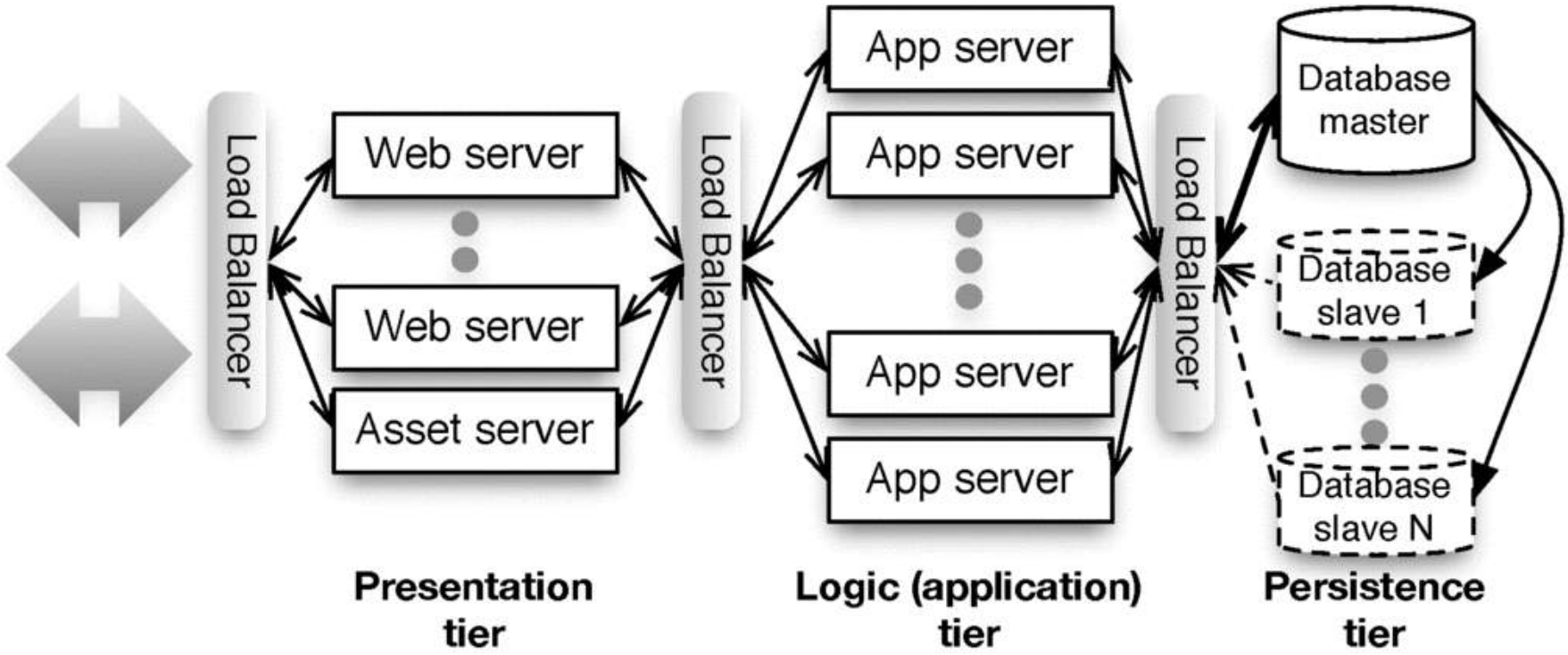


Developer

Medium-scale deployment

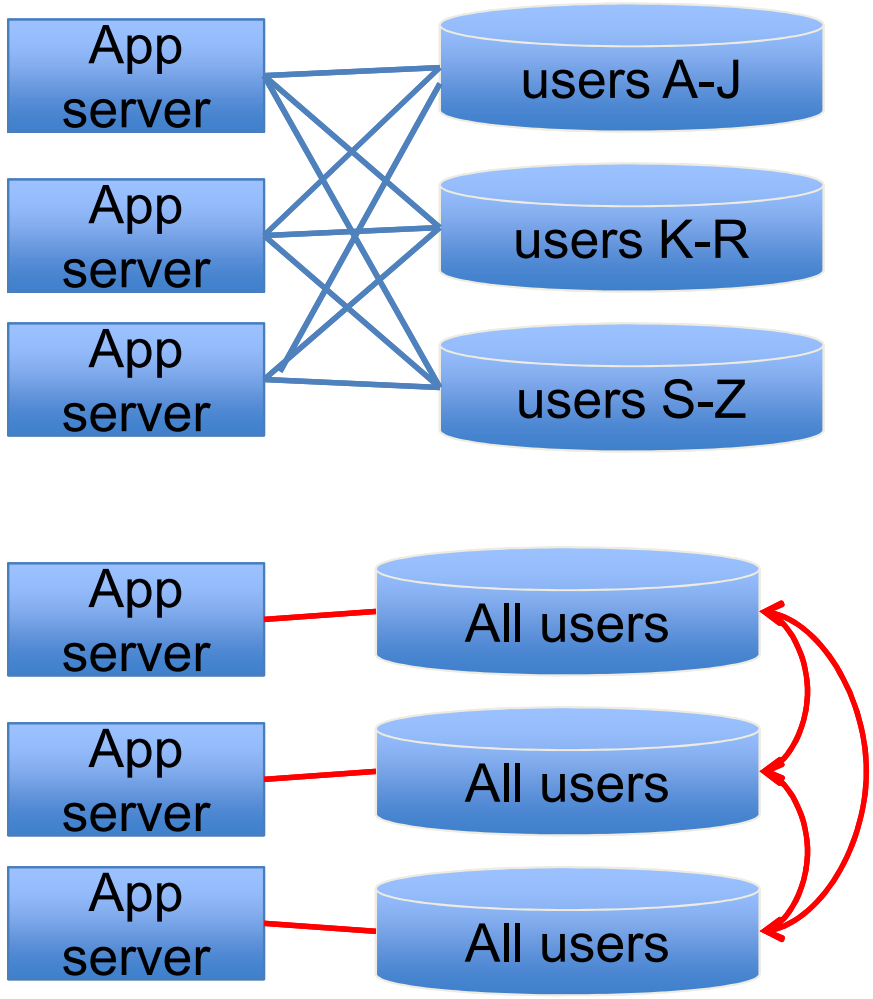
Large-scale curated deployment, e.g. Heroku

“Shared nothing”



Sharding vs. Replication

- Partition data across independent "shards"?
  - + Scales great
  - – Bad when operations touch >1 table
  - – Example use: user profile
- Replicate all data everywhere?
  - + Multi-table queries fast
  - – Hard to scale: writes must propagate to all copies => temporary *inconsistency* in data values
  - – Example: Facebook wall posts/"likes"



Summary: Web SaaS

- Browser *requests* web resource (URI) using HTTP
  - HTTP is a simple request-reply protocol that relies on TCP/IP
  - In SaaS, most URI's cause a program to be run, rather than a static file to be fetched
- *HTML* is used to encode content, *CSS* to style it visually
- *Cookies* allow server to track client
- JavaScript, TypeScript, …
- *Frameworks* make all these abstractions convenient for programmers to use, without sweating the details: Django, Angular, React/Redux
- Server-side frameworks help map SaaS to 3-tier, shared-nothing architecture