# Web Server Architecture

April 26, 2015

# Overview

Basics

Performance and scalability

Resource Planning

Web Server Architecture

# A Real Interview Question

What happens after you type a URL into a browser and press enter?

How does a browser work?

How does a web server work?

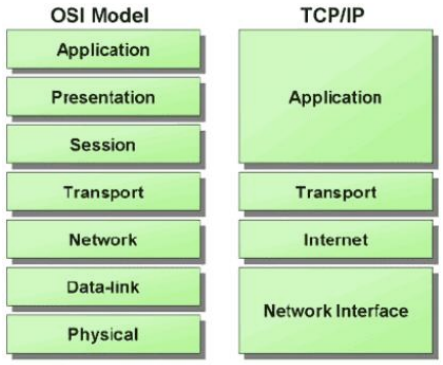
How do they connect?

What are URL, domain name, IP address, network port, HTTP, TCP connection, web, server, proxy, gateway, router, switch, cache, cookie, …

# The Web

The Internet protocol suite (TCP/IP protocol suite)

- TCP/IP: Provide reliable data transmission.
  - Socket interface:
    - bind(s, <local IP, port>), connect(s, <remote IP>, port)
    - listen(s), read(s, buffer, n), write(s, buffer, n), close(s)
- Application layer: (before the Web)
  - Telnet (remote terminal), SMTP, POP (email), FTP: file transfer



TCP/IP and the OSI model

Idea of Web: link **resources**

Key technologies (key problems to solve)

- How to identify resources: **URI** (by name (URN) or by location (URL))
- How to embed resource identifiers in resources: **HTML**
- How to transfer resources with embedded links: **HTTP**
- Of course, a client software (a browser) and a server software (HTTP server / web server)

How they became universal:

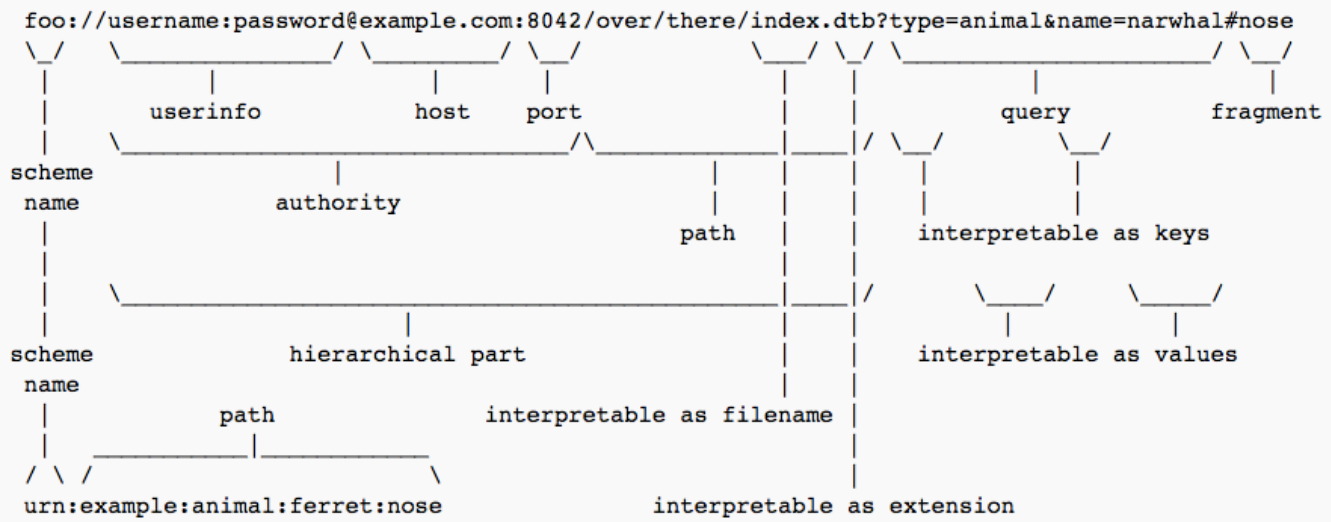
- URI -> Identifies not only linked resources (HTML), but also static files (css, js, image, video), dynamic apps, even in other protocols (ftp).
- HTML -> Embeds not only links (<a></a>), but also tables, forms, images, styles, scripts, etc.
- HTTP -> Transfer not only hypertext, but any resources.

# URL

Idea: machine location in internet + file location in machine

<scheme name> : **<hierarchical part>** [ ? <query> ] [ # <fragment> ]

Web server may interpret “path” and “query” arbitrarily.



# DNS

Map host names to IP addresses.

Essentially a distributed database system.

Return a list or return any one from a list, providing fault tolerance and load balancing.

History:

Directly use IP addresses

“hosts.txt” file stored on every computer

Steps:

DNS Cache, hosts.txt, Recursive DNS (provided by ISP), root DNS, Top-Level-Domain DNS, Authoritative DNS

More:

http://www.verisigninc.com/en_US/domain-names/online/how-dns-works/index.xhtml

Try:

$ nslookup google.com

$ nslookup www.google.com

$ nslookup maps.google.com

$ nslookup google.com

# HTTP

HTTP standard specifies the syntax and semantics of HTTP request and response.

How HTTP client should compose HTTP request and interpret HTTP responses.

How HTTP server should interpret HTTP request and compose HTTP response.

HTTP server may do anything in response to different HTTP methods

GET, HEAD, POST, PUT, DELETE, OPTION, TRACE

* like C++ standard specifies how C++ compiler interpret C++ code.

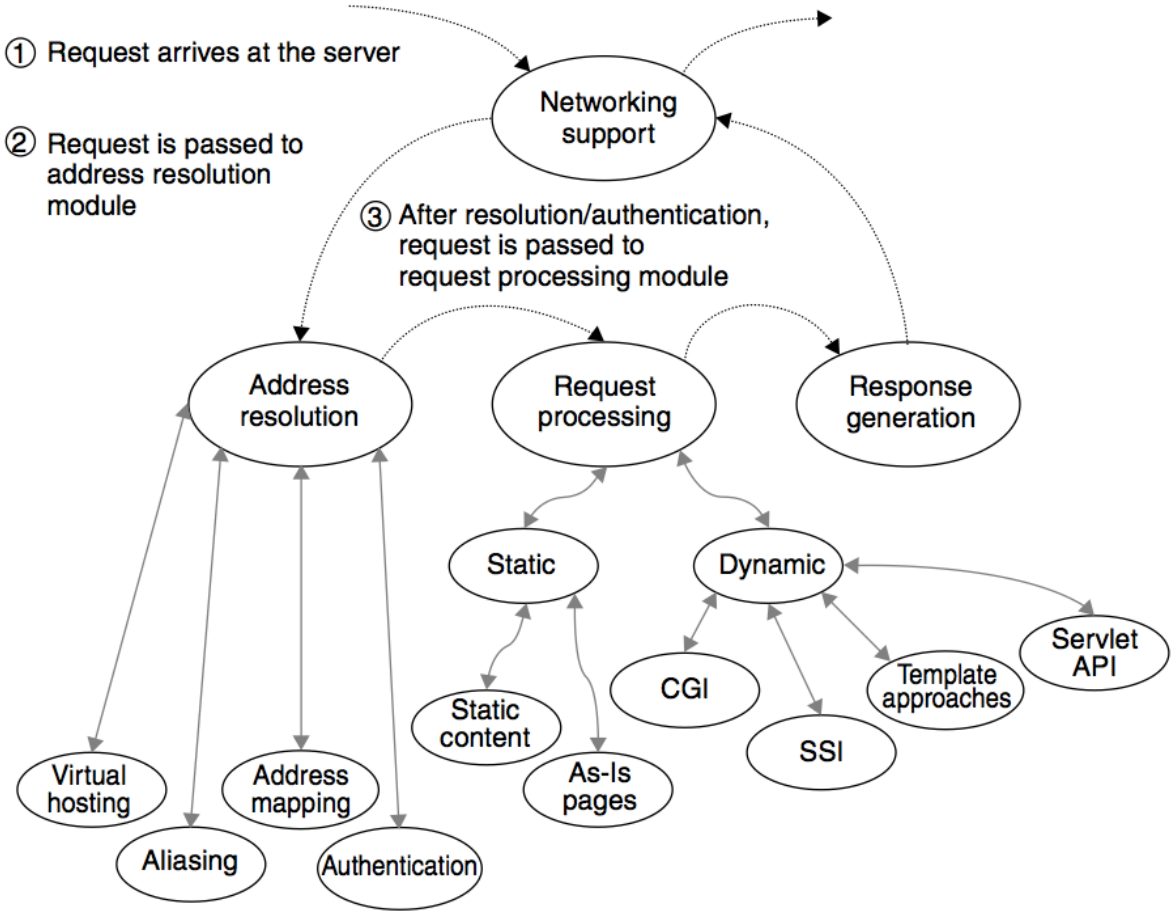
```
HTTP/version-number   status-code   message
Header-Name-1: value
Header-Name-2: value

[ response body ]
```

```
METHOD /path-to-resource  HTTP/version-number
Header-Name-1: value
Header-Name-2: value

[  optional request body  ]
```

```
GET /sj/index.html HTTP/1.1
Host: www.mywebsite.com
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 9934
 ...

<HTML>
<HEAD>
<TITLE>SJ's Web Page</TITLE>
</HEAD>
<BODY BGCOLOR="#ffffff">
<H2 ALIGN="center">Welcome to Sviergn Jiernsen's Home Page</H2>
 ...
</H2>
</BODY>
</HTML>
```

# Web Server Operations



# How Server Gets Request

A daemon (background process) listens (polling or interrupt) on the port 80 (a chunk of memory).

Once a TCP connection request arrives, the daemon process fork/spawns a new process/thread (or select a thread in a thread pool) for that TCP connection.

Each HTTP request will be in one TCP connection.

# *Virtual Hosting

* Can one machine (with one IP address) serve multiple domains?

    Yes. With virtual hosting.

    How to distinguish requests for domain A and requests for domain B? Their TCP destination are all 216.58.192.14:80

        Look at Host header (required by HTTP/1.1)

** One machine can also have multiple IP addresses (even with only one network card.)

    IP-based virtual hosting

# Further Learning

Compare HTTP and FTP for file transfer.

Compare TCP and UDP.

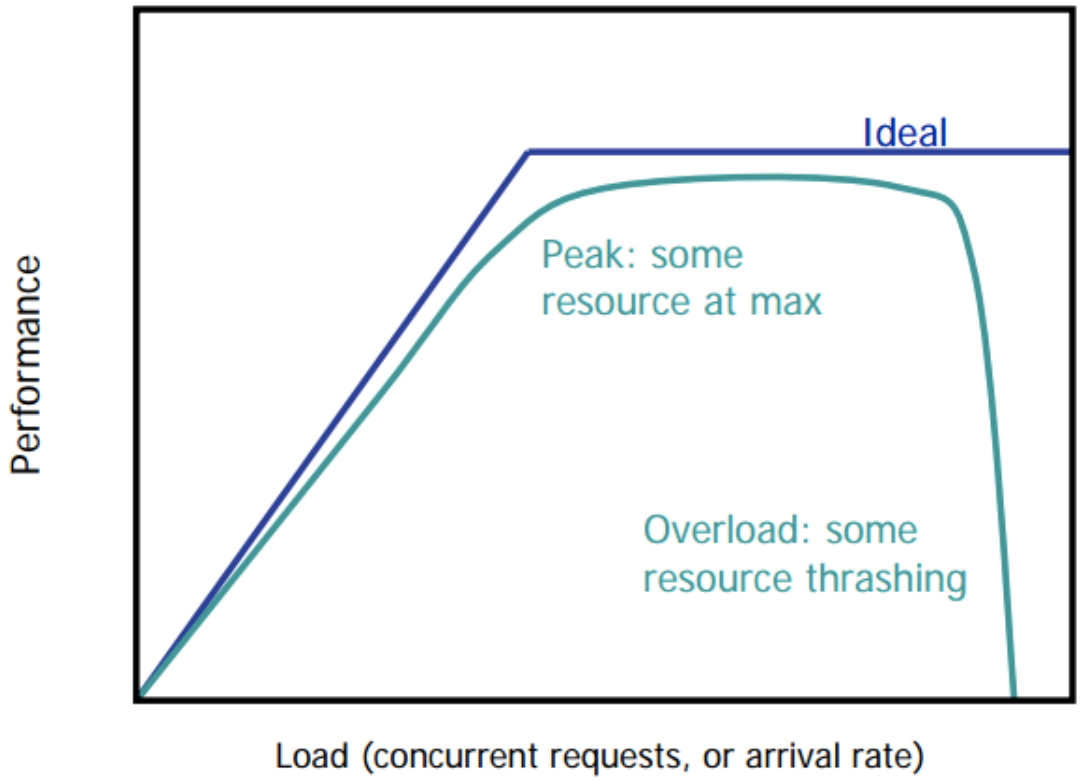
Compare IPv4 and IPv6.

What is HTTP cookie for?

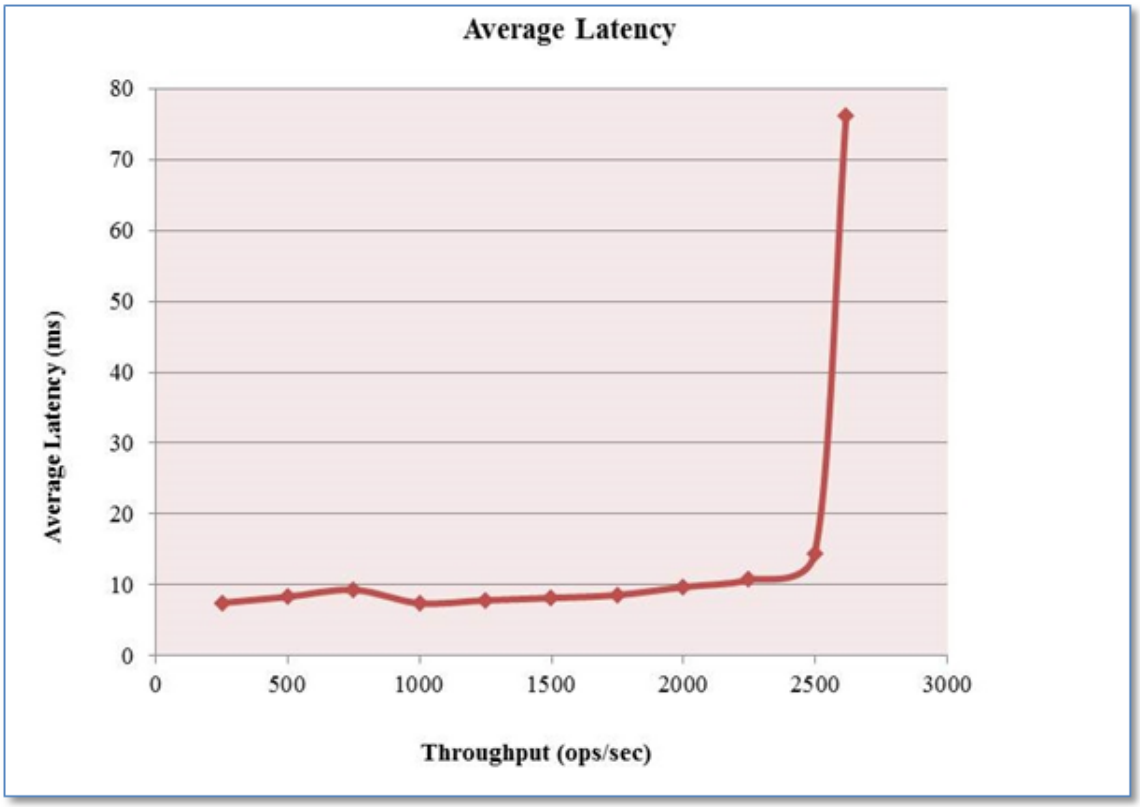
What is HTTP cache for?

What is HTTP proxy for?

# Throughput vs. Load



# Latency vs. Load

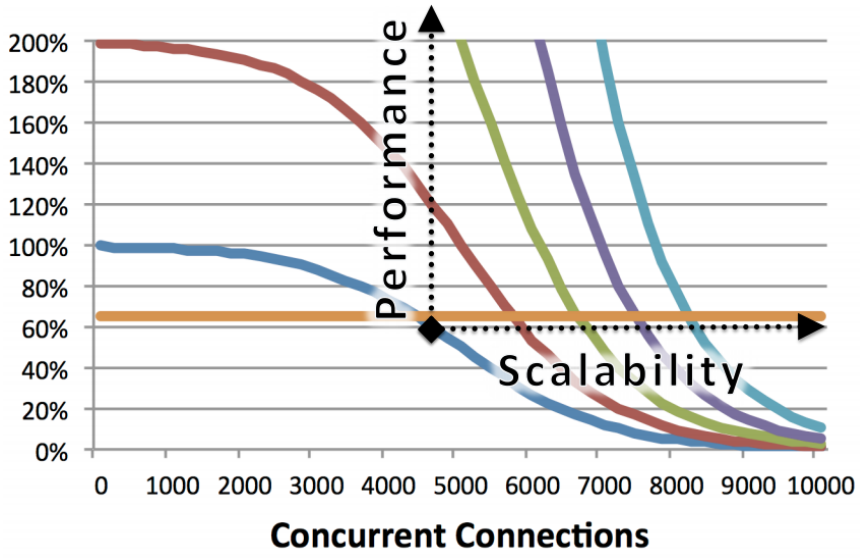


# *The C10K Problem

Performance is not only related to load (incoming QPS), but also to the number of concurrent connections.

1k QPS, 1s session -> 1k concurrent connections

1k QPS, 10s session -> 10 k concurrent connections



# *The C10K Problem

Reason:

n threads, m packets

Inside OS kernel, each thread checks if it should process that packet.

Overall complexity O(n*m)

Solution:

Optimize kernel

Event-driven solution: node.js

More on:

http://www.kegel.com/c10k.html

https://www.youtube.com/watch?v=73XNtI0w7jA

# Resource Planing

How many resources do you need for your services?

# Resource planning steps

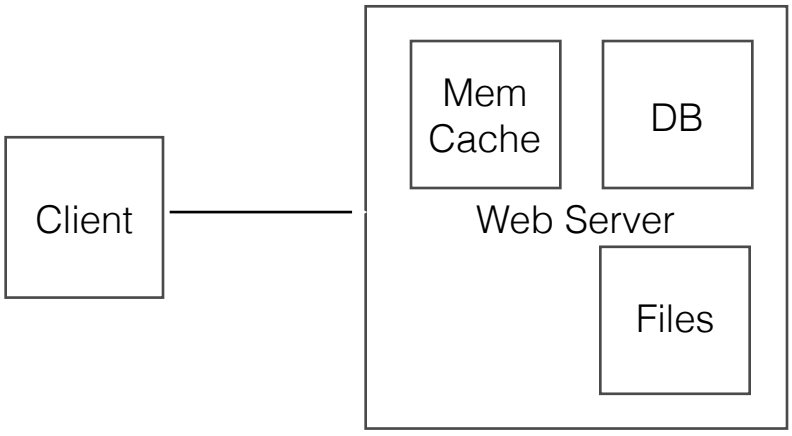
- Estimate CPU requirement
  - Estimate traffic (QPS)
    - For every level of server (reverse proxy, app server, backend server, data server), estimate the number of HTTP requests per second. (market analysis)
    - Factors: total number of users, online ratio, active ratio, how many HTTP requests per action, cache hit ratio
  - Estimate latency
    - Load testing
    - Factors: functionalities, cache hit ratio, network latency.
  - Estimate latency turning point (max QPS) for each node
    - Load testing
    - Factors: job latency, CPU usage per job, CPUs per node.
  - Allow for variations, overhead.
- Estimate RAM, storage requirement.
- Estimate network bandwidth requirement.
  - Factors: HTTP request/response size, QPS.

# Design Website Architecture

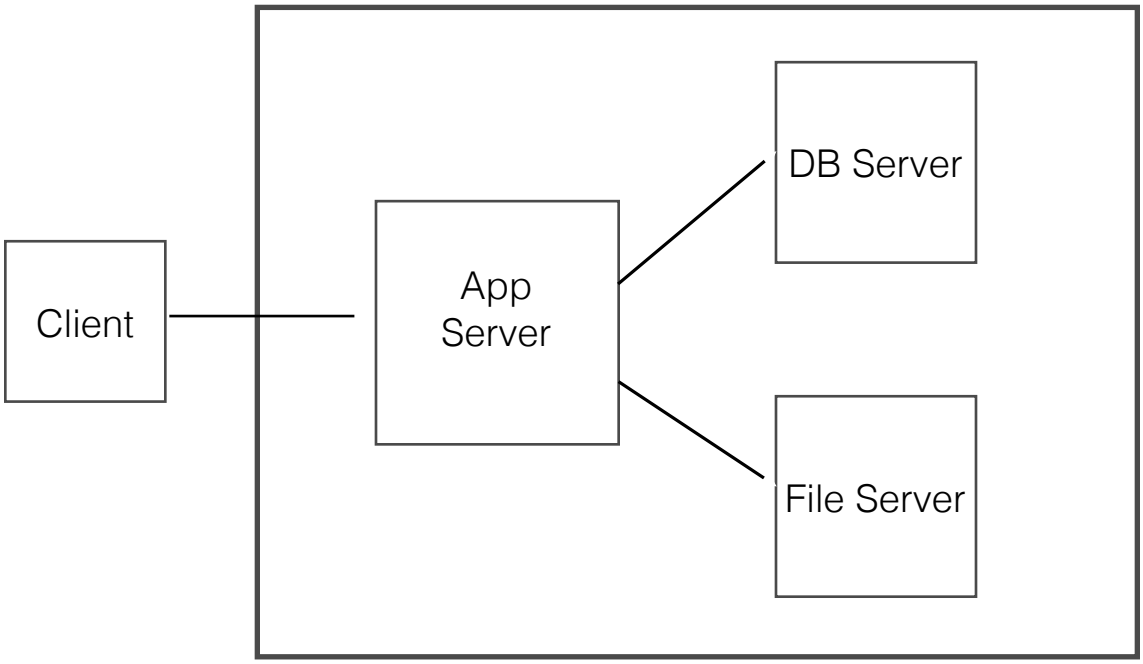
How would you design a website architecture?

e.g. Amazon, Youtube, Facebook, Google, Taobao

# Level 1



# Level 2



# Level 2

Why separating data server and app server?

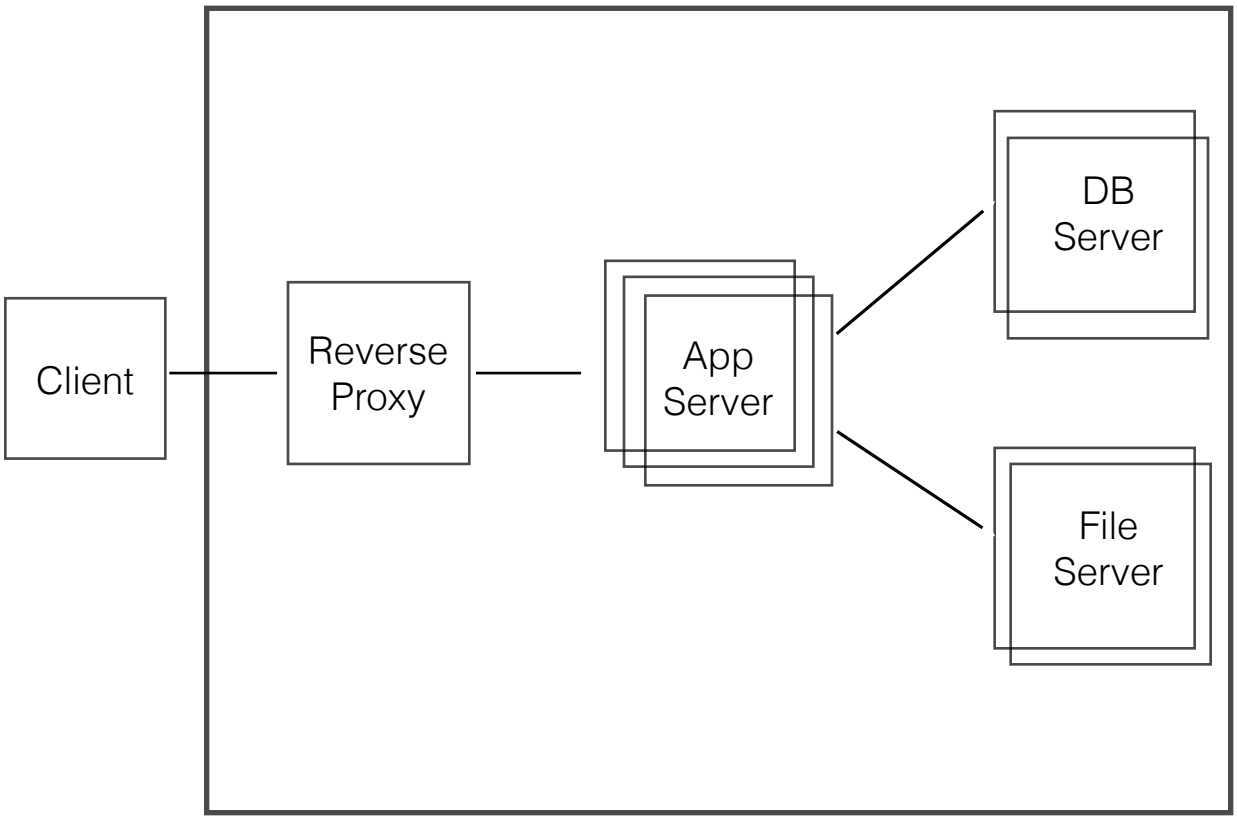
- Improves security: protect your database!
- Improves performance (not surprising, since we put more machines)
  - Resource contention is fewer.
- Reduces maintenance cost (also easier to find perf. bottleneck)
- Enables flexible numbers of DB servers for performance, reliability
- Con: network latency.

Add cache server when last level database is slow to access.

- Storage hierarchy.

Summary: decoupling and modularity is good.

# Level 3

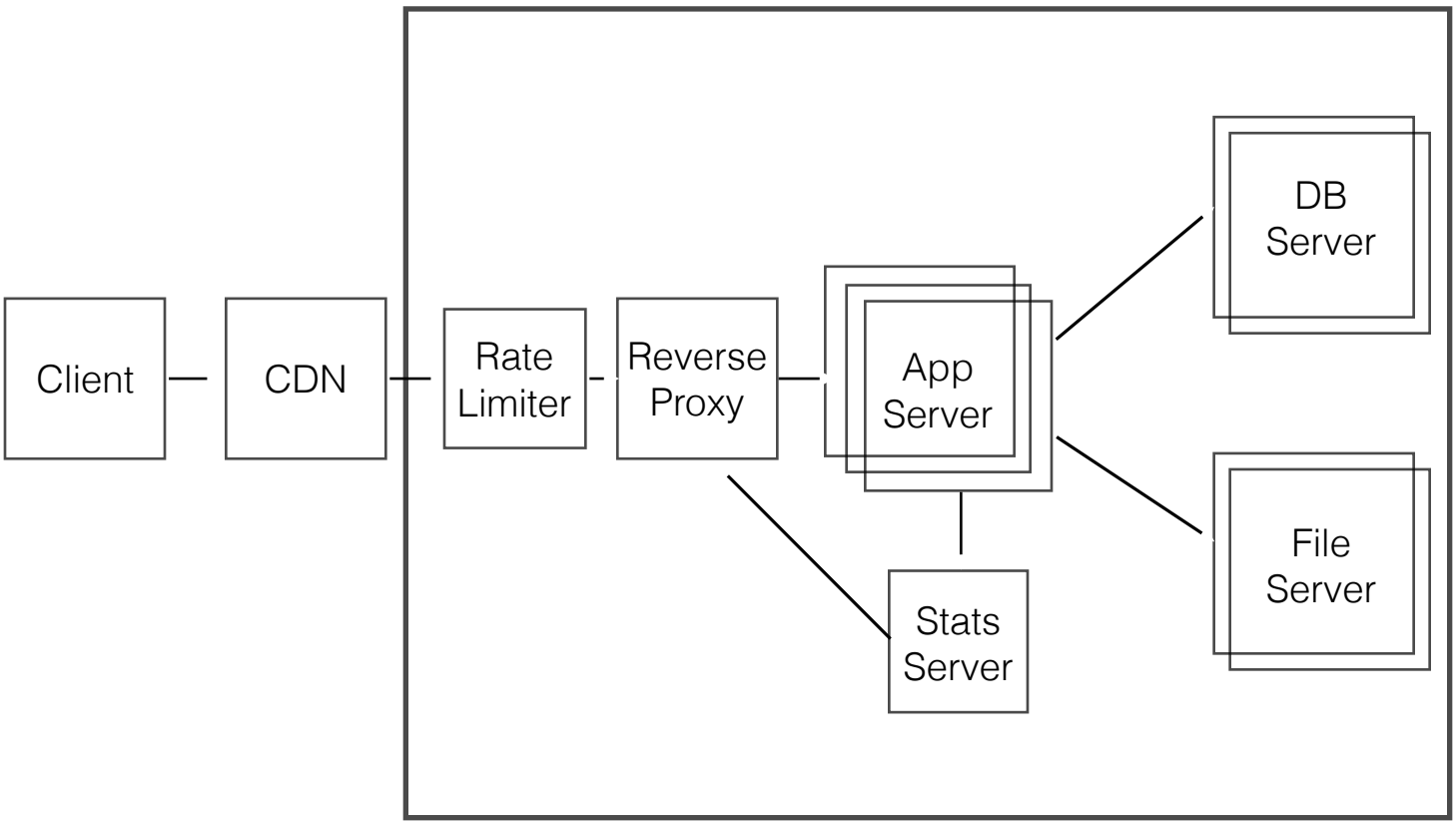


# Level 3

Why using reverse proxy?

- Improves security: Protect your app server!
- Load balancing is the key to horizontal scaling
  - Load balancing is at every level (beginning from DNS)
- Can cache static files.
  - Cache is at every level (beginning from browser)

# Level 4


Client
CDN
Rate Limiter
Reverse Proxy
App Server
DB Server
File Server
Stats Server

# Level 4

Content Delivery Network

- Put static content (videos, images, pdfs) near users. Reduce server load.

Use rate limiter to guarantee latency.

Use stats server to guide load balancing.

# References

[1] “HTTP The Definitive Guide”

[2] “Web Application Architecture”

# Further Watching

RESTful Web Service API

https://www.youtube.com/watch?v=7YcW25PHnAA

How We've Scaled Dropbox

https://www.youtube.com/watch?v=PE4gwstWhmc