# Computer Science E-75 Lecture 9 Scalability

- ☐ Questions
  - ☐ Do we have to respond to the window size changing?
    - ☐ No, if the project specs do not specify, there is no need to.
  - ☐ Google Maps makes Ajax calls to Google's servers; doesn't this violate Same-Origin Policy?
    - ☐ No, Google Maps uses an IFRAME loaded from Google's servers.
- ☐ Next week, guest lecture.
  - ☐ Advanced CSS lecture
- ☐ Scalability
  - ☐ If you have more users than you have for this course's projects, then you may run into problems with scalability.
  - ☐ David's favorite book: "High Performance MySQL"
- ☐ Vertical v. Horizontal Scaling
  - ☐ Vertical scaling: Throwing money at the problem. Bigger servers, more RAM, more CPUs. (One instances each of Apache, MySQL, etc.)
    - ☐ Advantage: Identical in design to what you might deploy for ten users.
    - ☐ Disadvantages
      - ☐ You have to throw money at the problem
      - ☐ Single point of failure
      - ☐ You can only go so far (There comes a point when you can't upgrade, and diminishing returns)
    - ☐ What can you do?
      - ☐ CPU: cores, L2 cache
      - ☐ Disk: PATA, SATA, SAS, ... / RAID
      - ☐ RAM
  - ☐ Horizontal scaling: Having redundancy, multiple (cheaper?) machines. Spreading them out. Distributed network. Clusters of systems. It's not just a matter of buying lots of machines and connecting them together, though. Multiple MySQL servers. Multiple Apache servers.
    - ☐ Using commodity hardware
- ☐ What can you do?
  - ☐ Write better code (optimize code so it runs faster)
  - ☐ PHP Acceleration
    - ☐ PHP is an interpreted language == slow (because scripts are not compiled; they're translated every single time they're run)
    - ☐ We can cache the compiled form (opcode) by installing modules that do so (PHP accelerators)
  - ☐ Load Balancing at Layer 4 (Transport layer)
    - ☐ What if you have three servers running the same software?
      - ☐ They all have to return the same results
      - ☐ We need a load balancer to decide which web server a given user's request should go for

**Computer Science E-75 Lecture 9 Scalability**

- DNS is a layer of indirection we already have
- Therefore, we can give the first server IP address 1.2.3.4, the second 1.2.3.5., etc., and get the DNS server to return each of these different IP addresses every time it is asked to return the address for, say, cs75.net.
- Very simply, we can simply give BIND (the DNS server that cs75.net runs) several A records. BIND will automatically give different IP addresses in round-robin format.
- Problems
  - Sessions; if you are logged into one server, then you're not logged into other servers.
  - If there is a database, you need to worry about transactions/locking (Greater contention for a database)
  - If one of the servers goes down, then some users will still get redirected to the dead server (DNS has Time To Live (TTL) of 1 hour, 3 hours, or longer. There are also caches between the users and the DNS server)

- Load Balancing at Layer 7 (Application layer)
  - A simple solution (Partitioning): Have half the alphabet go on one box, the other half go on another box. (You can have only half the database on each box) - Or you can just hash, say, the PHP session ID and decide which server to redirect the user to.
    - Problems
      - But now the load balancer is heavily loaded as it has to check each HTTP request header
      - There's still the dead server problem
      - What if some people use a lot of resources, but other people don't? Because the load balancer only knows about the number of people who have been redirected in the past (not how much load they are adding to each server), the partitioning is inefficient.
      - You can't refer to other parts of the database if databases are on other machines.
  - Sticky sessions (Sessions are stored on particular machines)
    - We can do Layer 7 Load Balancing (Each session still lives on one box, but the load balancer can figure out which machine each user's session lives on and redirect them there.)
    - Shared storage (Put the session files on something that's stored on one machine)
    - Using cookies (In an e-commerce site, you could store the shopping cart in cookies - put the burden on the user's browser)
      - Users can't now use the site if they don't have cookies
      - You can just store tokens in a cookie and look up tokens in a database
- Load Balancers
  - Software/Hardware solutions (Load balancers may automatically alert the sysops if a machine goes offline; they can find out the load of each machine.)
  - HTTPS is more computationally expensive; you need to think about whether it's worth using it or not if you have a large number of users

## Computer Science E-75 Lecture 9 Scalability

- ☐ Caching
  - ☐ If we think about the bottlenecks in our application (RAM is faster than disk, databases are slow)
  - ☐ Therefore, we can use database caching (by remembering answers to requests)
  - ☐ Web servers are extremely fast at sending HTML pages
    - ☐ Craigslist stores all their files as HTML (and periodically generates the HTML pages)
    - ☐ We can store parts of web pages (e.g. the CS75 course website's office hours notice on the right side of the page) in disk if generating the data is slow. Then we can read the cache in subsequent requests.
  - ☐ MySQL Query Cache (MySQL has a query cache that allows queries to be cached for quicker access later)
  - ☐ memcached (www.danga.com—memcached - www.php.net—memcache ) - We can store objects in memcached so we don't need to connect to the database every time.
    - ☐ First, connect to the memcached
    - ☐ If the object we're looking for (with a particular ID) exists in the cache:
      - ☐ Use that data
    - ☐ If not:
      - ☐ Connect to the database and fetch the data
      - ☐ Store that data in the cache so that subsequent requests can use it
  - ☐ MySQL
    - ☐ Layers:
      - ☐ Conneciton management, security (mysql_connect(), etc.)
      - ☐ SQL parsing, execution, caching, etc. (MySQL itself)
      - ☐ Storange engines (MyISAM - default, InnoDB - transactions, HEAP - in-memory, NDB - clustered): These actually store the data to a backend (usually a disk)
        - ☐ MyISAM has support for full-text search (Low disk use, Low memory use)
        - ☐ InnoDB has support for transactions, foreign key constraints (Low disk use, High memory use)
    - ☐ Replication: Master-Slave
      - ☐ If you own four computers with MySQL, you can configure one to be a "master", the other to be "slaves". Then you can turn on MySQL replication so that any changes that happen on the master machine propagate to the slaves. Therefore, any writes must be done to the master, but reading any of the slaves (which are read-only copies) you can get the same data.
    - ☐ Replication: Master-Master
      - ☐ You can have two masters, so that any writes that happen on one master get written to the other master as well.
    - ☐ You can even have Load Balancing + Replication (+ Partitioning)