Life After 50 (0:00 – 10:00)

Databases (10:00 – 19:30)

- Masters and slaves
  - Masters are main databases
  - Slaves are replications of masters
  - Typically, writes are done to masters and reads are done from slaves
  - Numbers of each reflect relative frequency of reads and writes
  - The company MySQL, for instance, has 1 master and 3 slaves because they expect more reads than writes
- Database administrator (DBA) – the person who runs designs the schema, etc., for a company's databases and tells the developers how to access them
- Memcache
  - Right now, every time one of your php pages is accessed in your pset 7, it is pulled up and interpreted anew, even if the same user accessed the same page just minutes before
  - Memcache is a piece of server software that caches the output of things like PHP files
  - For instance, on our website, the office hours page fetches data from the Google calendar only ever 60 seconds, and caches it for hits in between. This means that if when the calendar is updated it can take as much as 60 seconds for the results to appear on the website.
  - Of course this has drawbacks. If you made an update on your Facebook profile and didn't see the changes for 60 seconds, you wouldn't be a very happy user.

Maintaining a Website (19:30 – 32:00)

- How to put your website out there for people to view?
  - First, get your own domain name for $9.99 at a registrar (e.g. godaddy.com)
  - Get a webhost for a few dollars a month (e.g. Dreamhost)
  - Tell godaddy.com your domain name system
- Domain name system (DNS)
  - Servers that translate IP addresses into domain names and vice versa
  - Also informs the world of who your mail servers are
  - Find out more at computer.howstuffworks.com
- What if your website suddenly gets super popular?
- Our servers in the course can handle a few thousand hits per second
- More expensive servers can handle more
- But there are bottlenecks on the number of users you can tolerate per second:
  - Bandwidth
  - CPU
  - RAM

- So you can continue to get hardware with more RAM and faster CPUs. But there's also a ceiling on how much hardware you can throw at a problem
- Eventually, you have the best, most expensive hardware and may still have problems
- More robust solution: horizontal scaling

Horizontal Scaling (32:00 – 41:30)

- 4 of the cheapest end boxes instead of 1 of the most expensive
- Facebook, for instance, has 1800 database servers. If 1 goes down, it's not an issue.
- Lots of redundancy of data, resources to improve response time
- This is how Google manages to have such quick searches over such a huge amount of data
- How do you make use of servers when you have hundreds or servers, and each has many cores?
- One approach: virtualization
- Cloud computing is largely based on virtualization
- Take one server and chop it up into multiple servers, each of which can be sold/rented to someone who can use it with full administrative privileges
- Can also optimize performance through software
    - Mimimizing calls to expensive functions, like malloc()
    - Better hash functions, thinking about algorithms and data structures
    - Using bitwise operators
- Cache results of PHP compilation, so don't have to recompile every single time the same page is served up

Load Balancing (41:30 – 59:00)

- How to present multiple servers to the world as though they are 1? (e.g. so cnn.com can have multiples servers) DNS
- Have a bunch of servers with identical data, tell DNS one domain name maps to multiple IP addresses
- UNIX command nslookup will give us all of the IP addresses for a domain name
- When a user requests a domain name, they get any one of the IP addresses with equal probability
- What if a server goes down? Take it out of this list.
- Problem: bad IP address might be cached. Some percentage of users might hit a dead end.
- Solution: Layer 7 load balancing
    - For instance, on Facebook, it used to be that the URL would be different for each network. There was harvard.facebook.com, mit.facebook.com, etc.
    - In this situation, users could be sent to different servers based on the URL.

- Another problem: there is a need for shared storage that is not fulfilled when different users are sent to different servers
- One solution: file sharing. For our website, there is a directory called scratch that is accessible to all the servers and contains, for instance, shopping cart contents for each user
    - Problems with this solution: slower, single point of failure

Caching (59:00 – 71:30)

- Serving up static HTML is much faster than processing PHP that accesses SQL database
- Craglist uses HTML as a sort of cache. After generating the page, it is just stored as an HTML file.
- The result of this is that there can be a few hours lag in changes to Craigslist pages.
- One question to consider when managing a website that accesses a database is whether reads or writes are more common
    - In problem set 8, reads are more common, because you only do write when you run import that one initial time
- MySQL has a tool to cache query results for a certain amount of time in case the same query is made soon after
- Memcached is another freely available tool to implement cache for you
- Having masters and slaves allows you take advantage of lots of cheap hardware by having redundancy in data. Also allows you to take advantage of frequency differential between reads and writes
- Problem: single point of failure if you only have one master
- Solution: implement redundancy, have twice as much hardware
- Tradeoff: consistency issues and lag time because you have to copy data from one database to another
- See slides 28-32 for possible topologies