Distributed Systems Intro

Logistics

- Course Policies
  - see web page...
  - http://www.cs.cmu.edu/~dga/15-440/F11
  - obligatory discussion of {late days, cheating, etc.}
- Waitlist?
- No recitations this year
- Office hours (hands): Earlier, 3:30 - 5:30, 4:00 - 6:00, Sunday?

Waitlist

- Waitlist of unprecedented size. Keep coming to class, because we don't really know how it will work out.
- 74 registered
- 68 waitlisted. (!) (5 dupes. If you are on both, please remove from one. You know who you are. It will not increase admissions odds. :))
- The bad news: Not everyone will get in.
- The plea: Not serious about the class? DROP SOON.
- The strategy:
  - Attend class! Sign the signup sheet.
  - Let us know if class is on immediate graduation path; *have your academic advisor email us this. :-)* (this works well)
- Priority order for 440: SCS, CMU ugrads, others; 640: SCS MS, others

Course Goals

- Systems requirement:
  - Learn something about distributed systems in particular;
  - Learn general systems principles (modularity, layering, naming, security, ...)
  - Practice implementing real, larger systems; in teams; must run in nasty environment;
- One consequence: Must pass homeworks, exams, and projects independently as well as in total.

Course Format

- ~30 lectures
- Office hours: Practical issues for implementing projects; general questions and discussion
- 3 projects; 1 solo, 2 team
  - Distributed (internet-wide) password cracker
  - Building Tribbler
  - TBA: Either choose-your-own or Data-intensive cluster applications with MapReduce/Hadoop?

Book

- Link to Amazon purchase (new, used, rent) from syllabus page
- Several useful references on web page
- We'll be compiling notes (and these slides) for your use over the course of the semester; based on, but not identical to, prior 15-440 instance

About Projects

- Systems programming somewhat different from what you've done before
  - Low-level (C)
  - Often designed to run indefinitely (error handling must be rock solid)
  - Must be secure - horrible environment
  - Concurrency
  - Interfaces specified by documented protocols
- Office Hours & "System Hacker's View of Software Engineering"
  - Practical techniques designed to save you time & pain

Collaboration

- Working together important
  - Discuss course material
  - Work on problem debugging
- Parts *must* be your own work
  - Homeworks, midterm, final, solo proj
- Team projects: both students should understand entire project
- What we hate to say: we run cheat checkers...
- Partner problems: *address early*.

Late Work

- 10% penalty per day
- Can't be more than 2 days late
- Usual exceptions: documented medical, emergency, etc.
  - *Talk to us early if there's a problem!*
- Two "late points" to use - one day each (still can't be more than 2 days late)
- Regrade requests in writing to course admin

Why take this course?

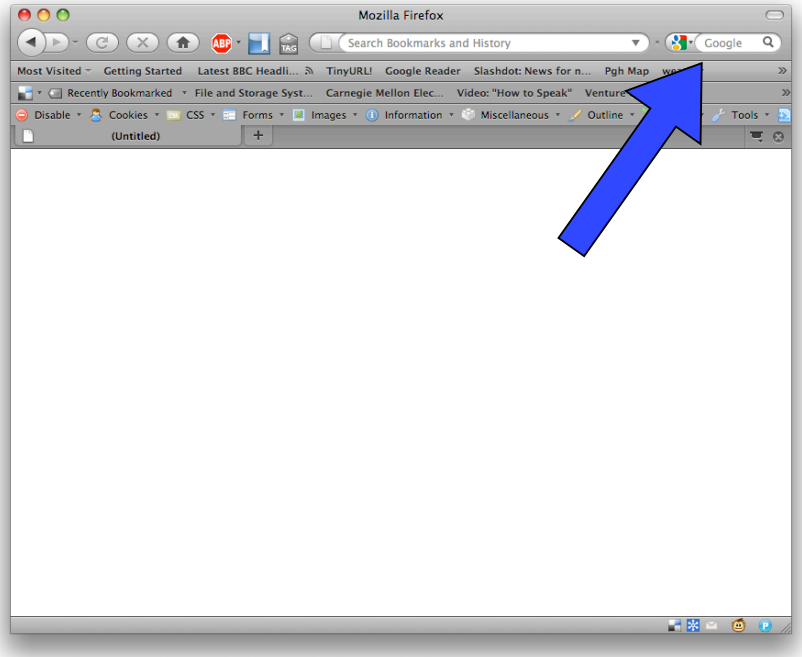
- Huge amounts of computing are now distributed...
  - A few years ago, Intel threw its hands up in the air: couldn't increase GHz much more without CPU temperatures reaching solar levels
  - But we can still stuff more transistors (Moore's Law)
  - Result: Multi-core and GPUs.
  - Result 2: Your computer has become a parallel/distributed system. In a decade, it may have 128 cores.
- Oh, yeah, and that whole Internet thing...
  - my phone syncs its calendar with google, which i can get on my desktop with a web browser, ...
    - (That phone has the computing power of a desktop from 10 years ago and communicates wirelessly at a rate 5x faster than the average american home could in 1999.)
  - Stunningly impressive capabilities now seem mundane. But *lots* of great stuff going on under the hood...
  - Most things are distributed, and more each day

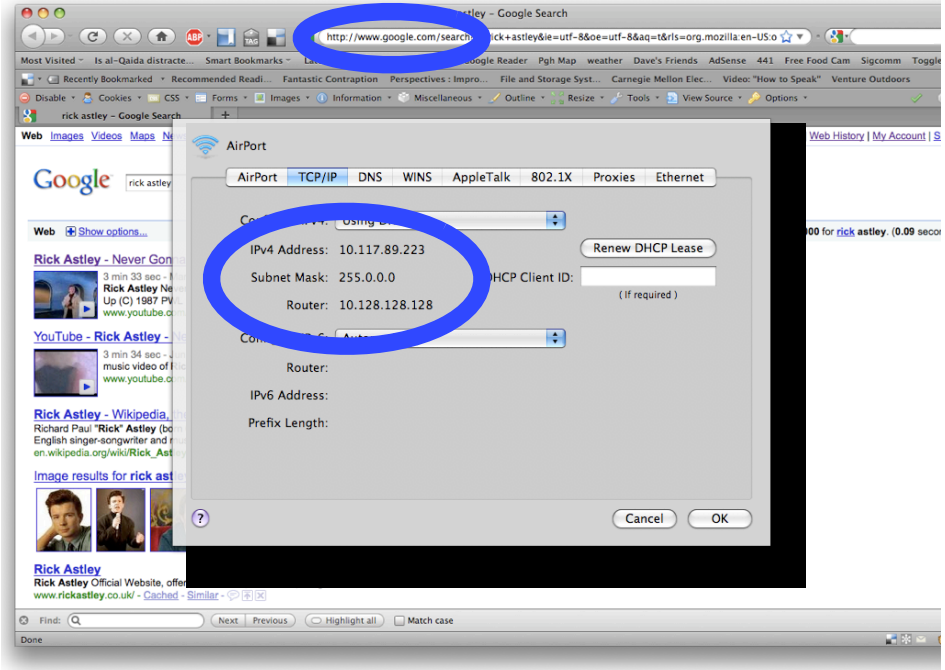
If you find yourself ...

- In hollywood....
  - ... rendering videos on clusters of 10s of 1000s of nodes?
  - Or getting terabytes of digital footage from on-location to post-processing?
- On wall street...
  - tanking our economy with powerful simulations running on large clusters of machines
  - For 11 years, the NYSE ran software from cornell systems folks to update trade data
- In biochem...
  - using protein folding models that require supercomputers to run
- In gaming...
  - Writing really bad distributed systems to enable MMOs to crash on a regular basis
- not to mention the obvious places

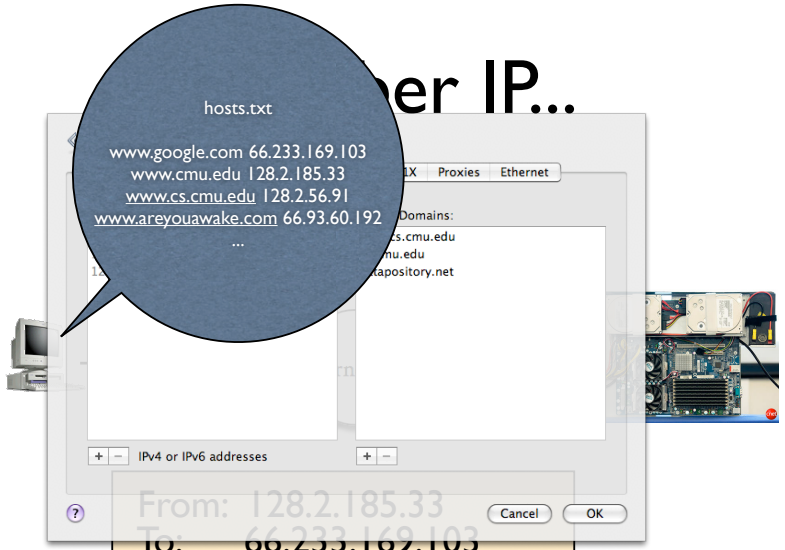
Enough advertising

- Let's look at one real distributed system
- That's drastically more complex than it might seem from the web browser...



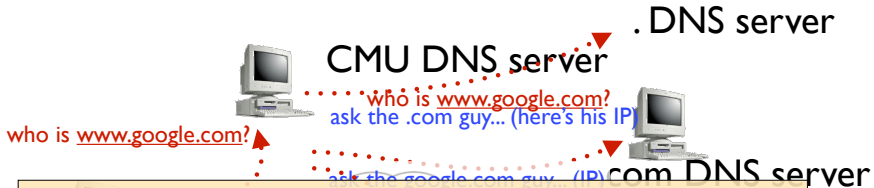


...er IP...



From: 128.2.185.33
To: 66.233.169.103
<packet contents>

Domain Name System

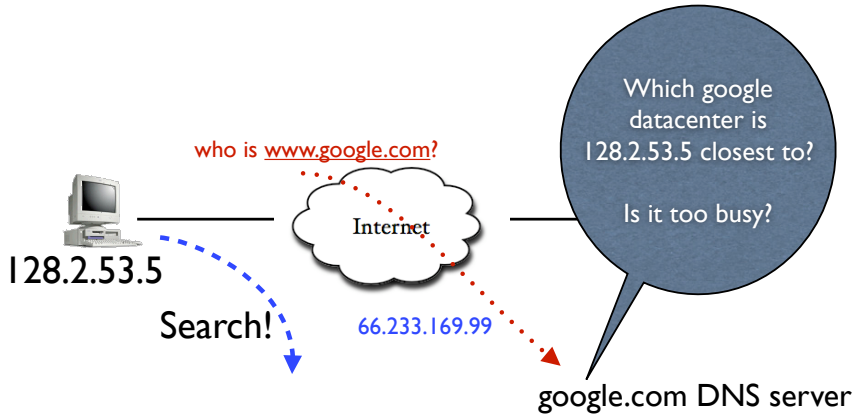


Decentralized - admins update own domains without coordinating with other domains

Scalable - used for hundreds of millions of domains

Robust - handles load and failures well

But there's more...

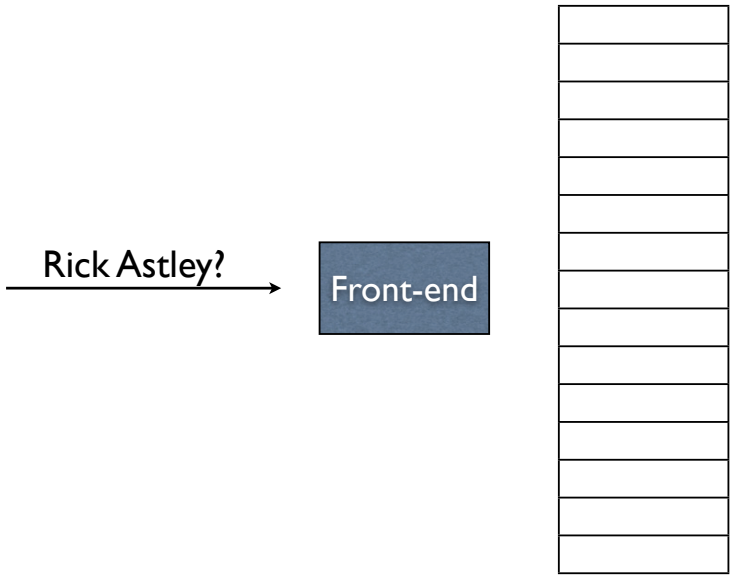


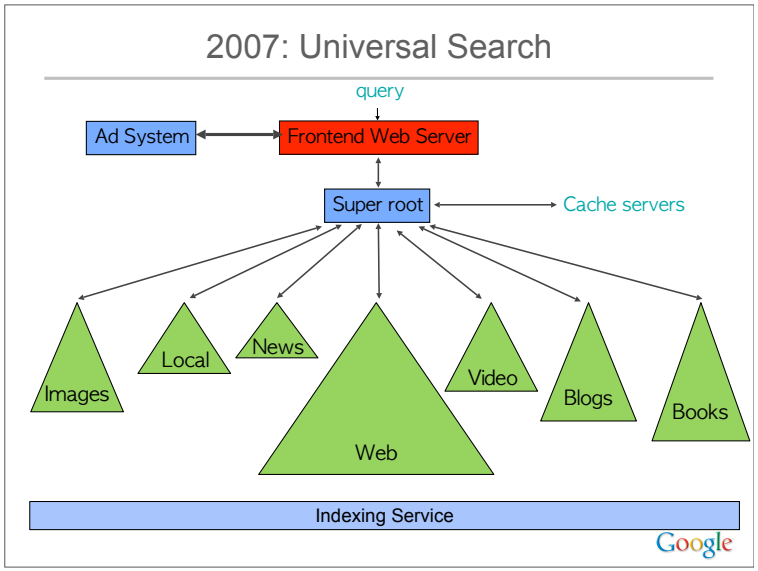
A Google Datacenter

How big? Perhaps one million+ machines

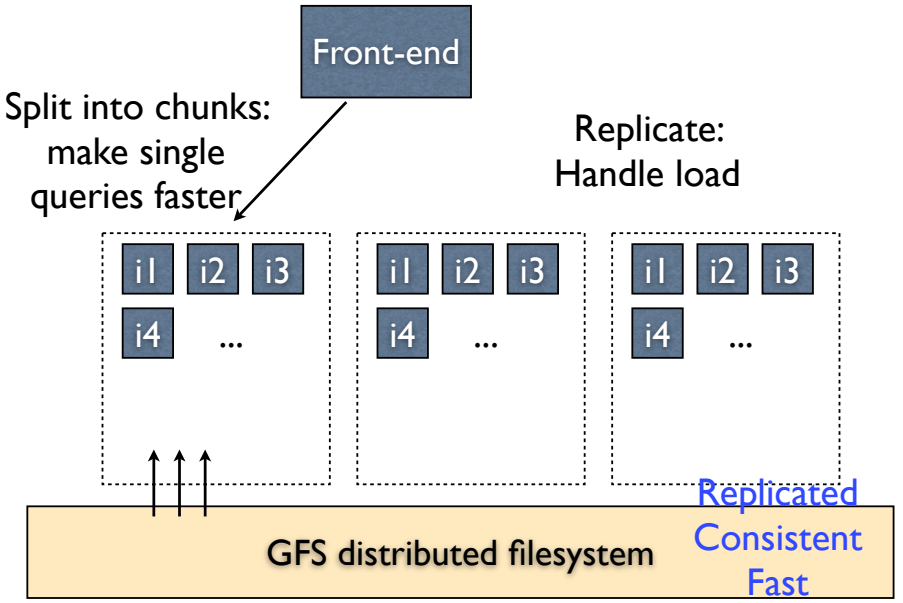
but it's not that bad...

usually don't use more than **20,000** machines to accomplish a single task. [2009, probably out of date]





slide from Jeff Dean, Google



How do you index the web?

1. Get a copy of the web.
2. Build indexes.
3. Profit.

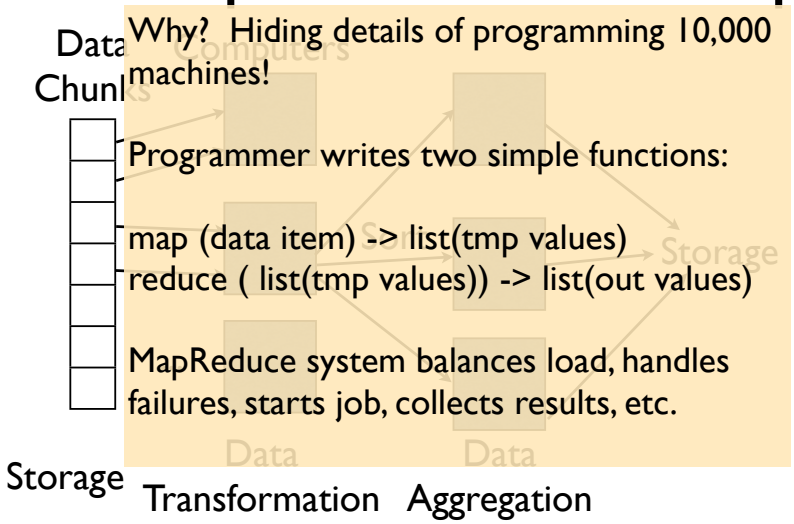
There are over 1 trillion unique URLs
Billions of unique web pages
Hundreds of millions of websites
30?? terabytes of text

=

- *Crawling* -- download those web pages
- *Indexing* -- harness 10s of thousands of machines to do it
- *Profiting* -- we leave that to you.

- *"Data-Intensive Computing"*

MapReduce / Hadoop



Why? Hiding details of programming 10,000 machines!

Programmer writes two simple functions:

map (data item) -> list(tmp values)
reduce (list(tmp values)) -> list(out values)

MapReduce system balances load, handles failures, starts job, collects results, etc.

All that...

- Hundreds of DNS servers
- Protocols on protocols on protocols
- Distributed network of Internet routers to get packets around the globe
- Hundreds of thousands of servers
- ... to find one bad video in under 1/2 second

How do you index the web?