Cloud Computing Lecture #1

What is Cloud Computing?

(and an intro to parallel/distributed processing)

Jimmy Lin
The iSchool
University of Maryland

Wednesday, September 3, 2008

Some material adapted from slides by Christophe Bisciglia, Aaron Kimball, & Sierra Michels-Slettvet, Google Distributed Computing Seminar, 2007 (licensed under Creation Commons Attribution 3.0 License)

Source: http://www.free-pictures-photos.com/

What is Cloud Computing?

1. Web-scale problems
2. Large data centers
3. Different models of computing
4. Highly-interactive Web applications

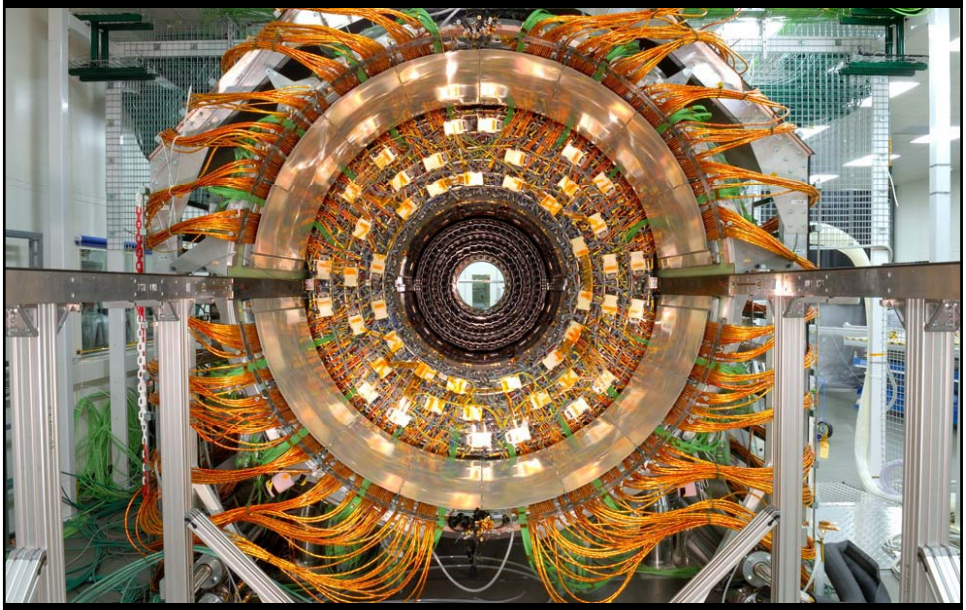
1. Web-Scale Problems

- Characteristics:
  - Definitely data-intensive
  - May also be processing intensive
- Examples:
  - Crawling, indexing, searching, mining the Web
  - "Post-genomics" life sciences research
  - Other scientific data (physics, astronomers, etc.)
  - Sensor networks
  - Web 2.0 applications
  - …

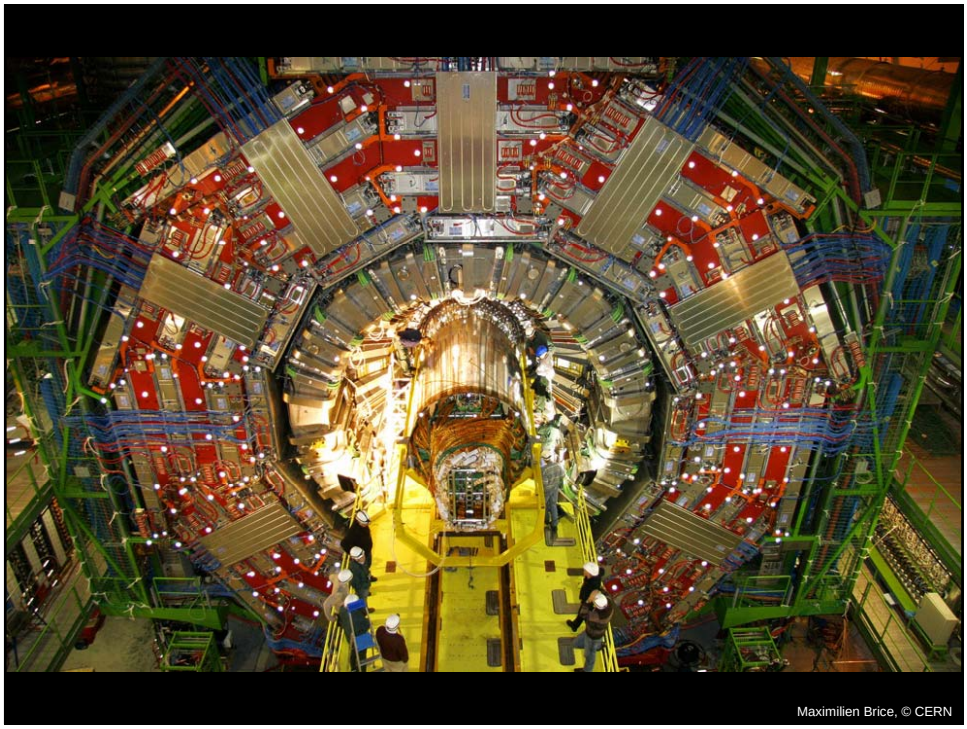
How much data?

- Wayback Machine has 2 PB + 20 TB/month (2006)
- Google processes 20 PB a day (2008)
- “all words ever spoken by human beings” ~ 5 EB
- NOAA has ~1 PB climate data (2007)
- CERN’s LHC will generate 15 PB a year (2008)

The iSchool
University of Maryland



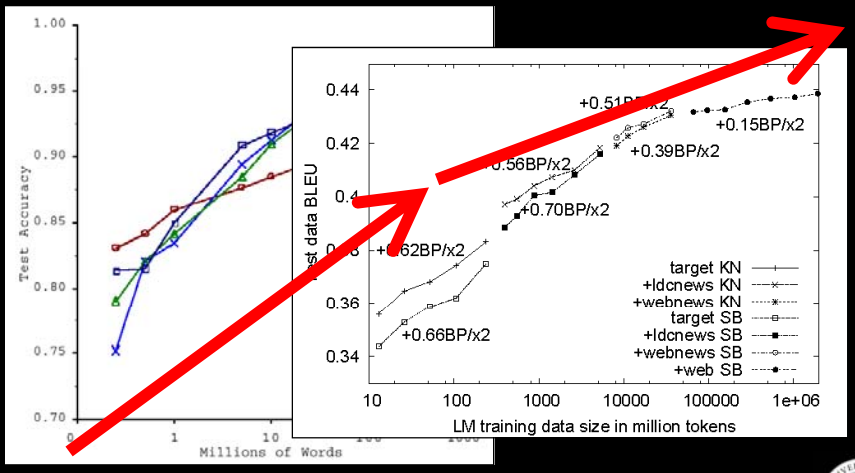
Maximilien Brice, © CERN



Maximilien Brice, © CERN

There's nothing like more data!

s/inspiration/data/g;



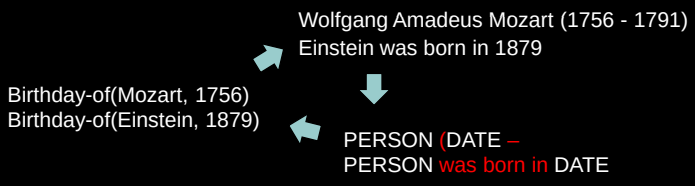
(Banko and Brill, ACL 2001)
(Brants et al., EMNLP 2007)

The iSchool
University of Maryland

What to do with more data?

- Answering factoid questions
  - Pattern matching on the Web
  - Works amazingly well

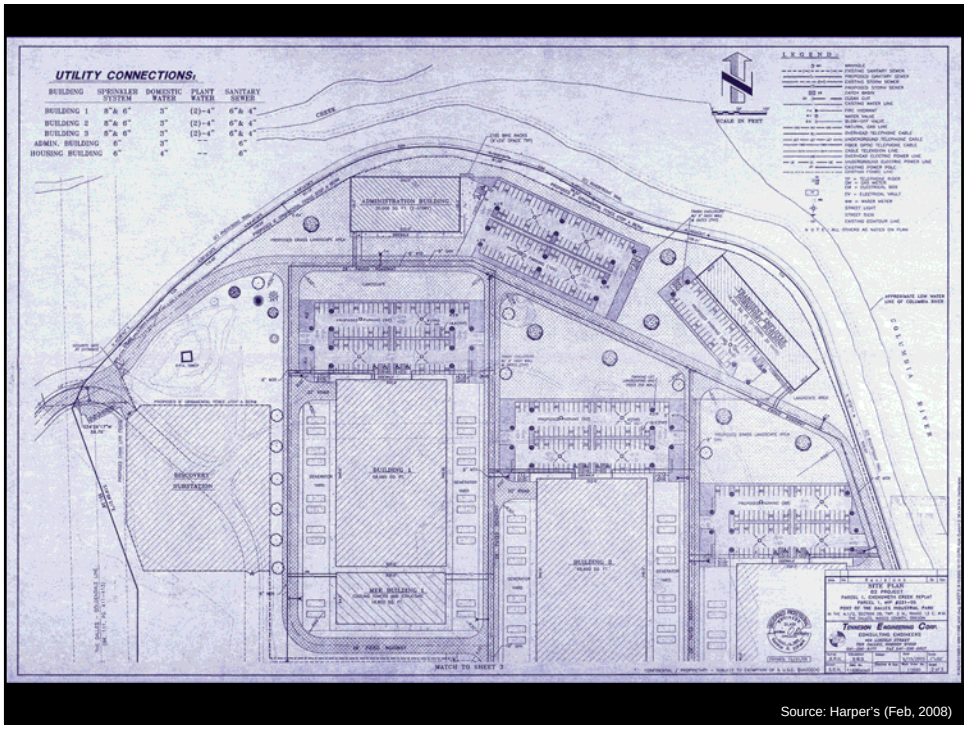
    Who shot Abraham Lincoln? → X shot Abraham Lincoln
- Learning relations
  - Start with seed instances
  - Search for patterns on the Web
  - Using patterns to find more instances



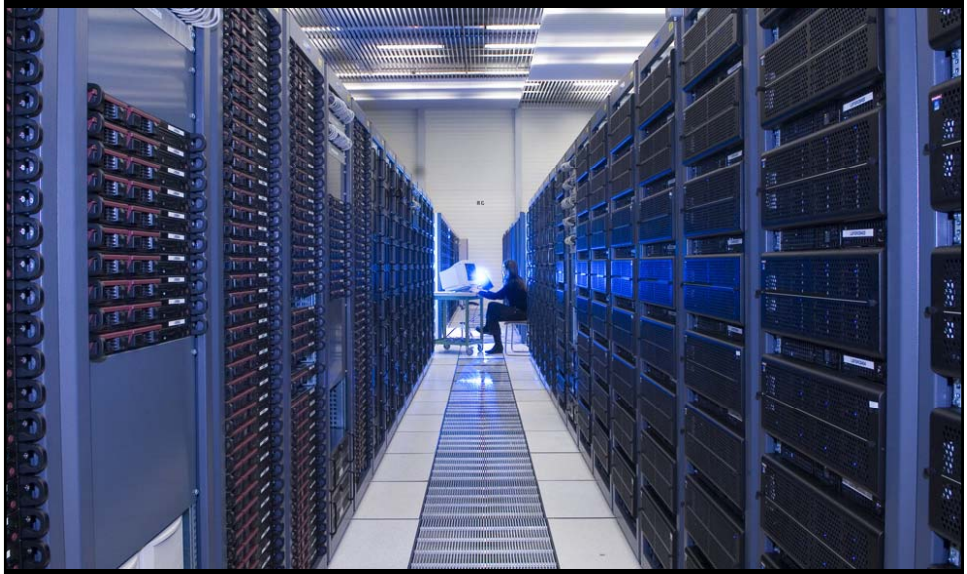
(Brill et al., TREC 2001; Lin, ACM TOIS 2007)
(Agichtein and Gravano, DL 2000; Ravichandran and Hovy, ACL 2002; …)

2. Large Data Centers

- Web-scale problems? Throw more machines at it!
- Clear trend: centralization of computing resources in large data centers
  - Necessary ingredients: fiber, juice, and space
  - What do Oregon, Iceland, and abandoned mines have in common?
- Important Issues:
  - Redundancy
  - Efficiency
  - Utilization
  - Management

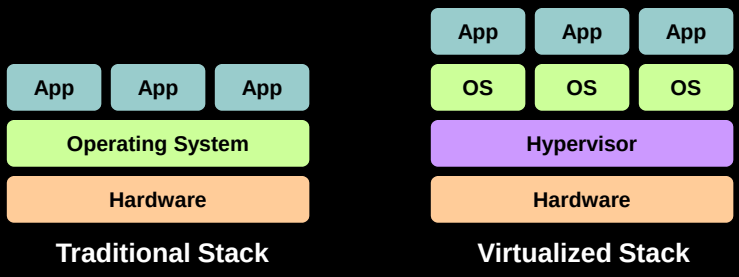


Source: Harper's (Feb, 2008)



Maximilien Brice, © CERN

Key Technology: Virtualization



The iSchool
University of Maryland

3. Different Computing Models

"Why do it yourself if you can pay someone to do it for you?"

- Utility computing
  - Why buy machines when you can rent cycles?
  - Examples: Amazon's EC2, GoGrid, AppNexus
- Platform as a Service (PaaS)
  - Give me nice API and take care of the implementation
  - Example: Google App Engine
- Software as a Service (SaaS)
  - Just run it for me!
  - Example: Gmail

The iSchool
University of Maryland

4. Web Applications

- A mistake on top of a hack built on sand held together by duct tape?
- What is the nature of software applications?
  - From the desktop to the browser
  - SaaS == Web-based applications
  - Examples: Google Maps, Facebook
- How do we deliver highly-interactive Web-based applications?
  - AJAX (asynchronous JavaScript and XML)
  - For better, or for worse…

What is the course about?

- MapReduce: the “back-end” of cloud computing
  - Batch-oriented processing of large datasets
- Ajax: the “front-end” of cloud computing
  - Highly-interactive Web-based applications
- Computing “in the clouds”
  - Amazon’s EC2/S3 as an example of utility computing

Amazon Web Services

- Elastic Compute Cloud (EC2)
  - Rent computing resources by the hour
  - Basic unit of accounting = instance-hour
  - Additional costs for bandwidth
- Simple Storage Service (S3)
  - Persistent storage
  - Charge by the GB/month
  - Additional costs for bandwidth
- You'll be using EC2/S3 for course assignments!

This course is not for you...

- If you're not genuinely interested in the topic
- If you're not ready to do a lot of programming
- If you're not open to thinking about computing in new ways
- If you can't cope with uncertainly, unpredictability, poor documentation, and immature software
- If you can't put in the time

Otherwise, this will be a richly rewarding course!

Cloud Computing Zen

- Don't get frustrated (take a deep breath)…
  - This is bleeding edge technology
  - Those W$*#T@F! moments
- Be patient…
  - This is the ~~second~~ first time I've taught this course
- Be flexible…
  - There will be unanticipated issues along the way
- Be constructive…
  - Tell me how I can make everyone's experience better


Source: Wikipedia


Source: Wikipedia

Source: Wikipedia

Source: Wikipedia

Things to go over…

- Course schedule
- Assignments and deliverables
- Amazon EC2/S3

The iSchool
University of Maryland

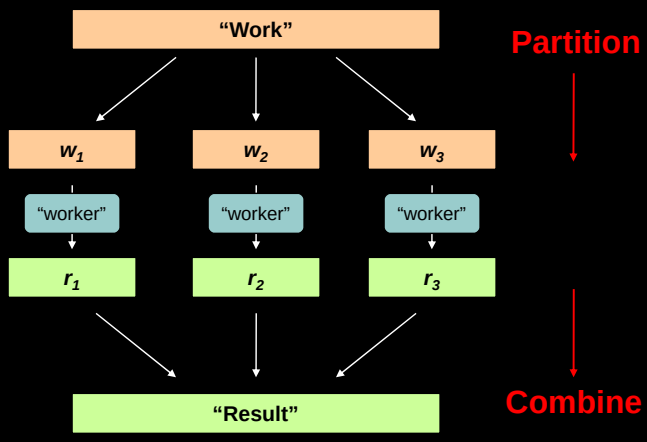
Web-Scale Problems?

- Don't hold your breath:
  - Biocomputing
  - Nanocomputing
  - Quantum computing
  - …
- It all boils down to…
  - Divide-and-conquer
  - Throwing more hardware at the problem

Simple to understand… a lifetime to master…

The iSchool
University of Maryland

Divide and Conquer



The iSchool
University of Maryland

Different Workers

- Different threads in the same core
- Different cores in the same CPU
- Different CPUs in a multi-processor system
- Different machines in a distributed system

The iSchool
University of Maryland

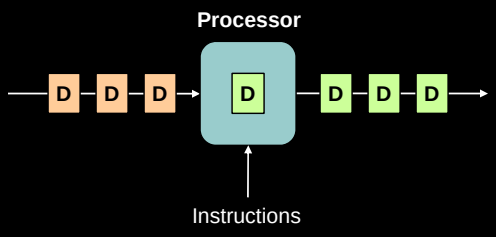
Choices, Choices, Choices

- Commodity vs. “exotic” hardware
- Number of machines vs. processor vs. cores
- Bandwidth of memory vs. disk vs. network
- Different programming models

Flynn’s Taxonomy

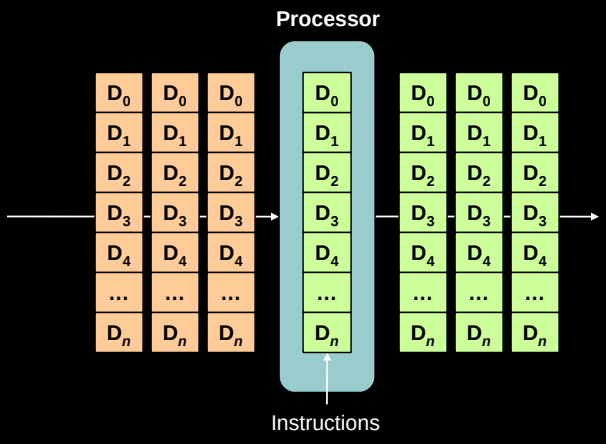
Data \ Instructions	Single (SI)	Multiple (MI)
Single (SD)	**SISD** Single-threaded process	**MISD** Pipeline architecture
Multiple (MD)	**SIMD** Vector Processing	**MIMD** Multi-threaded Programming

SISD



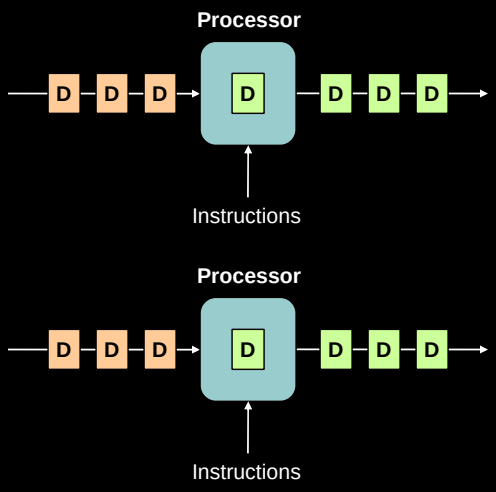
The iSchool
University of Maryland

SIMD



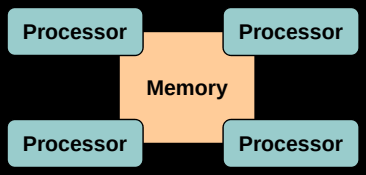
The iSchool
University of Maryland

MIMD



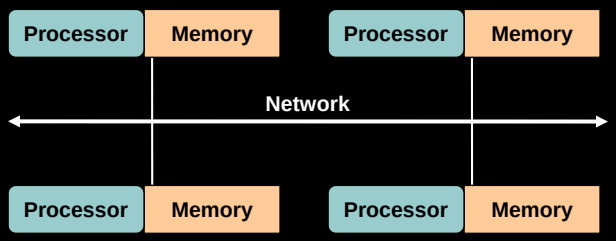
The iSchool
University of Maryland

Memory Typology: Shared

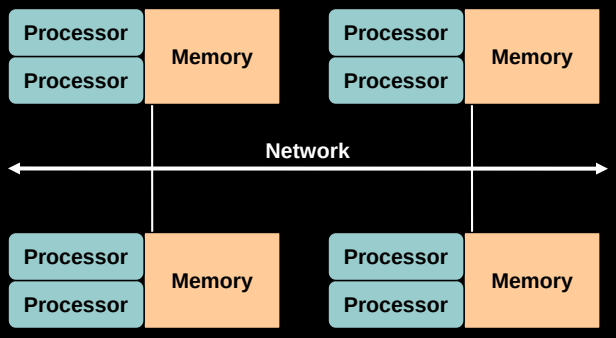


The iSchool
University of Maryland

Memory Typology: Distributed



Memory Typology: Hybrid



Parallelization Problems

- How do we assign work units to workers?
- What if we have more work units than workers?
- What if workers need to share partial results?
- How do we aggregate partial results?
- How do we know all the workers have finished?
- What if workers die?

What is the common theme of all of these problems?

The iSchool
University of Maryland

General Theme?

- Parallelization problems arise from:
  - Communication between workers
  - Access to shared resources (e.g., data)
- Thus, we need a synchronization system!
- This is tricky:
  - Finding bugs is hard
  - Solving bugs is even harder

The iSchool
University of Maryland

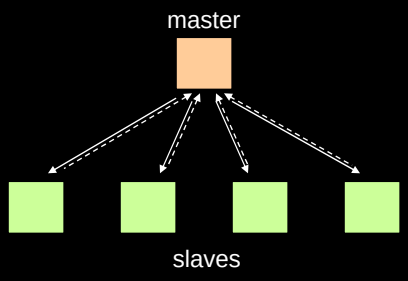
Managing Multiple Workers

- Difficult because
  - (Often) don't know the order in which workers run
  - (Often) don't know where the workers are running
  - (Often) don't know when workers interrupt each other
- Thus, we need:
  - Semaphores (lock, unlock)
  - Conditional variables (wait, notify, broadcast)
  - Barriers
- Still, lots of problems:
  - Deadlock, livelock, race conditions, ...
- Moral of the story: be careful!
  - Even trickier if the workers are on different machines

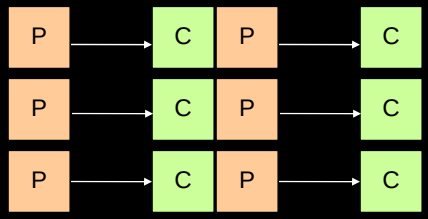
Patterns for Parallelism

- Parallel computing has been around for decades
- Here are some "design patterns" …

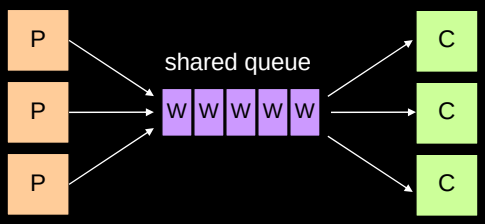
Master/Slaves



Producer/Consumer Flow



Work Queues



Rubber Meets Road

- From patterns to implementation:
  - pthreads, OpenMP for multi-threaded programming
  - MPI for clustering computing
  - …
- The reality:
  - Lots of one-off solutions, custom code
  - Write you own dedicated library, then program with it
  - Burden on the programmer to explicitly manage everything
- MapReduce to the rescue!
  - (for next time)