What is cloud computing? Why is this different?

Jimmy Lin
The iSchool
University of Maryland

Monday, March 30, 2009

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
  - Data warehouses
  - Sensor networks
  - “Post-genomics” life sciences research
  - Other scientific data (physics, astronomy, etc.)
  - Web 2.0 applications
  - ...

How much data?
- Internet archive has 2 PB of data + 20 TB/month
- Google processes 20 PB a day (2008)
- “all words ever spoken by human beings” ~ 5 EB
- CERN’s LHC will generate 10-15 PB a year
- Sanger anticipates 6 PB of data in 2009

640K ought to be enough for anybody.
There’s nothing like more data!

s/inspiration/data/g;

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
    - Wolfgang Amadeus Mozart (1756 – 1791)
    - Einstein was born in 1879
    - Birthday-of(Mozart, 1756)
    - Birthday-of(Einstein, 1879)
    - PERSON DATE
    - PERSON was born at DATE

How do I make money?

- Petabytes of valuable customer data...
  - Sitting idle in existing data warehouses
  - Overflowing out of existing data warehouses
  - Simply being thrown away

- Source of data:
  - OLTP
  - User behavior logs
  - Call-center logs
  - Web crawls, public datasets
  - ...

- Structured data (today) vs. unstructured data (tomorrow)

- How can an organization derive value from all this data?
2. Large Data Centers
- Web-scale problems? Throw more machines at it!
- Centralization of resources in large data centers
  - Necessary ingredients: fiber, juice, and land
  - What do Oregon, Iceland, and abandoned mines have in common?
- Important Issues:
  - Efficiency
  - Redundancy
  - Utilization
  - Security
  - Management overhead

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

4. Web Applications
- What is the nature of future 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)
  - A hack on top of a mistake built on sand, all held together by duct tape and chewing gum?
  - For better, or for worse…
What is the course about?
1. Web-scale problems
2. Large data centers
3. Different models of computing
4. Highly-interactive Web applications

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...

Divide and Conquer

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

Haven’t we been here before?
(Quick tour through parallel and distributed computing)

Flynn’s Taxonomy

Instructions
<table>
<thead>
<tr>
<th>Single (SI)</th>
<th>Multiple (MI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SISD</td>
<td>MISP</td>
</tr>
<tr>
<td>single-threaded process</td>
<td></td>
</tr>
<tr>
<td>pipeline architecture</td>
<td></td>
</tr>
<tr>
<td>SIMD</td>
<td>MIMD</td>
</tr>
<tr>
<td>vector processing</td>
<td></td>
</tr>
<tr>
<td>multi-threaded processes</td>
<td></td>
</tr>
</tbody>
</table>
Choices, Choices, Choices
- Commodity vs. "exotic" hardware
- Scale "up" or scale "out"
- Number of machines vs. processor vs. cores
- Bandwidth of memory vs. disk vs. network
- Different programming models

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?

General Theme?
- Parallelization problems arise from:
  - Communication between workers
  - Access to shared resources
- Thus, we need a synchronization system!
- This is tricky:
  - Finding bugs is hard
  - Solving bugs is even harder

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!
“Design Patterns”

Rubber, meet road…
Rubber, Meet Road

- **Existing tools:**
  - pthreads, OpenMP for multi-threaded programming
  - MPI for clustering computing
  - Condor, PBS, SGE, etc. for higher-level job management

- **The reality:**
  - Lots of one-off solutions, custom code
  - Write your own dedicated library, then program with it
  - Burden on the programmer to explicitly manage everything

What's different now?

Questions?