Cloud Computing

CMSC 433 Spring 2012

Based on slides prepared by Prof. Jimmy Lin
What is Cloud Computing?

- No single definition, but some common elements include:

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

- **Characteristics:**
  - Data-intensive
  - May also be processing intensive

- **Examples:**
  - Crawling, indexing, searching, mining the Web
  - Large-scale scientific simulation & data processing
    - physics, astronomy, etc.
  - Sensor networks
    - Weather modeling
  - Web 2.0 applications
  - Intelligence analysis
    - Analyzing financial transactions
How much data is there?

Google: processes 20 PB a day (2008)

eBay: 6.5 PB of user data + 50 TB/day (5/2009)

Wayback Machine: 3 PB + 100 TB/month (3/2009)

LHC: 15 PB a year (any day now)

Facebook: 36 PB of user data + 80-90 TB/day (6/2010)

LSST: 6-10 PB a year (~2015)

How much data is there?
Data Sets are Growing

How do we get here if we’re not Google?

(Banko and Brill, ACL 2001)
(Brants et al., EMNLP 2007)
2. Large Data Centers

- Trend: centralize computing resources in large data centers
- Exploits relatively cheap hardware
  - Scale out, not up
- Location criteria include:
  - Fiber
  - Power
  - Water
  - Space
  - Proximity to other data centers
UTILITY CONNECTIONS:

<table>
<thead>
<tr>
<th>BUILDING</th>
<th>SPRINKLER SYSTEM</th>
<th>DOMESTIC WATER</th>
<th>PLANT WATER</th>
<th>SANITARY WASTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUILDING 1</td>
<td>8&quot; &amp; 3&quot;</td>
<td>3&quot;</td>
<td>(2) 6&quot;</td>
<td>6&quot; &amp; 6&quot;</td>
</tr>
<tr>
<td>BUILDING 2</td>
<td>8&quot; &amp; 6&quot;</td>
<td>3&quot;</td>
<td>(2) 6&quot;</td>
<td>6&quot; &amp; 6&quot;</td>
</tr>
<tr>
<td>BUILDING 3</td>
<td>8&quot; &amp; 6&quot;</td>
<td>3&quot;</td>
<td>(2) 6&quot;</td>
<td>6&quot; &amp; 6&quot;</td>
</tr>
<tr>
<td>ADMIN. BUILDING</td>
<td>6&quot;</td>
<td></td>
<td></td>
<td>6&quot;</td>
</tr>
<tr>
<td>HOUSING BUILDING</td>
<td>6&quot; &amp; 4&quot;</td>
<td></td>
<td></td>
<td>6&quot;</td>
</tr>
</tbody>
</table>
3. Different Computing Models

Buy vs. Build

- Software as a Service (SaaS)
  - Buy use of an application
    - Gmail

- Platform as a Service (PaaS)
  - Buy access to an API
    - Google App Engine

- Infrastructure as a Service (IaaS)
  - Buy computing cycles
    - Amazon’s EC2, GoGrid
One example: 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
4. Web Applications

- Often accessed from desktop via a browser
  - Often have Web-based UIs
  - Examples: Google Maps, Facebook

- How do we deliver highly-interactive Web-based applications?
  - AJAX (asynchronous JavaScript and XML)
Key Technology: Virtualization

Traditional Stack:
- Hardware
- Operating System
- App

Virtualized Stack:
- Hardware
- Hypervisor
- Operating System
- App
Key Algorithmic Approach: Divide and Conquer

Partition

Merge

"Work"

"worker"

"worker"

"worker"

"Result"
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
Choices, Choices, Choices

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

- Basic patterns are the same as what we’ve discussed in this course

- Examples include:
  - Master/Slave
  - Producer/Consumer
  - Work Queues
Master/Slaves
Producer/Consumer Flow
Work Queues
Concurrent Challenges in the Cloud

- Concurrency is difficult to reason about
  - At the scale of datacenters (even across datacenters)
  - In the presence of failures
  - In terms of multiple interacting services

- Lots of experimentation with tools and languages
  - Lots of one-off solutions, custom code
  - Write your own dedicated library, then program with it
  - Burden on the programmer to explicitly manage everything
Continuing Challenges

- Finding the right level of abstraction
- Hiding system-level details from the developers
  - No more race conditions, lock contention, etc.
- Separating the *what* from the *how*
  - Developer specifies the computation that needs to be performed
  - Execution framework (“runtime”) handles actual execution
“Big Ideas”

- Scale “out”, not “up”
  - Limits of SMP and large shared-memory machines
- Move processing to the data
  - Cluster have limited bandwidth
- Process data sequentially, avoid random access
  - Seeks are expensive, disk throughput is reasonable
- Seamless scalability
  - From the mythical man-month to the tradable machine-hour
Web-Scale Problems

- For now, cloud computing approaches emphasize
  - Divide-and-conquer
  - Commodity computing devices

cheap commodity clusters
+ simple, distributed programming models
= data-intensive computing for the masses!