Cloud Computing & Fintech

Mendel Rosenblum

Stanford University CS349f - 4/22/2019
Agenda

● Cloud Computing Enabling Technologies
  ○ Virtualization
  ○ Scale-out architecture

● Cloud Computing Offerings
  ○ Infrastructure as a service
  ○ Platform as a service
  ○ Software as a service

● Fintech computing and the Cloud
  ○ Deliberate control
  ○ Reflex control
Virtualization
Virtualization according to the dictionary

... the act of creating a virtual version of something

**Virtual:** not physically existing as such but made by software to appear to do so

"All problems in computer science can be solved by another level of indirection" - David Wheeler

Except too many levels of indirection
Computing prior to virtualization: Compute

- **Computing prior to virtualization:**
  - **Compute CPUs, Memory, I/O Devices**
  - **Export abstractions:** Files, Processes, etc.
  - **Manage hardware resources**

**Functionality**

- App #1
- App #2
- App #3
- App #4

**Operating System**

- Export abstractions: Files, Processes, etc.
- Manage hardware resources

**Hardware**

- CPUs, Memory, I/O Devices

**Source**

- Various vendors
- Windows /x86 Bundle - WinTel
Computing prior to virtualization: Networks

Machine to Machine using network interface addresses
Computing prior to virtualization: Storage

- File Servers

- Storage Area Networks
Big Idea: Binding part of software to hw is bad

- Can run software any hardware - not just one machine
  - Handle failures, load balancing, etc.
- Problem: No one else saw or wanted this

<table>
<thead>
<tr>
<th>App #1</th>
<th>App #2</th>
<th>App #3</th>
<th>App #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUs, Memory, I/O Devices</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export abstractions: Files, Processes, etc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage hardware resources</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>App #1</th>
<th>App #2</th>
<th>App #3</th>
<th>App #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUs, Memory, I/O Devices</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Export abstractions: Files, Processes, etc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage hardware resources</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Virtual Machine Monitors - Special OS-like software

- Used on 1960s IBM mainframes: Rare computer: Multiple Virtual Machines

![Diagram of Virtual Machines and OSes]

Export abstractions: hardware resources (Virtual CPUs, Virtual Memory, Virtual I/O Devices)
Manage hardware resources (CPUs, Memory, I/O Devices)
Virtual Networks

- Tell each VM which VMs it can talk to

Can do either in VMM or in network switches (VLANs)
Virtual Machines give us the binding we want

- Can treat software (OS + Apps - Functionality) as first-class entity
Small irony: IBM rediscovered VMMs

1999: IBM wanted to sell big x86 machines (Sequent): Server Consolidation
Scale-out Architecture
Scale-out architecture

- An alternative to buying a bigger machine (scale-up architecture)
- Spread work across a collection of machines
Scale-out challenges

- Dividing up work across multiple machines is hard - difficult programming
  - Balance load, minimize communication, determining best scale
  - Dealing with failures and stragglers

- Could use a large number of machines for short periods of time
  - Does not match old-style computer buying pattern

But many of the challenges are not application-specific:
  - Amenable to system software infrastructure
Scale-out: Bulk processing with Map/Reduce

● Key idea: Map/Reduce infrastructure handles all scale-out issues
  ○ Manages load balancing, data distribution, failure and strangler handing
  ○ Operation: allocate computing resources for Map and Reduce processing
    i. Bring in data across the Map processors
    ii. Run Map function on each Map processor
    iii. Optimized shuffle of data to the Reduce processors
    iv. Run the Reduce function on each Reduce processor

● Greatly simplified programming model for scale-out
  ○ Users only specify two functions: Map and Reduce

● Does limit to problems decomposable to map/reduce

● Newer approach with less programming restrictions: Spark
Scale-out: Web/Mobile App Backend

Internet

Load Balancer

Web Server

Web Server

Web Server

Web Server

Web Server

Web Server

Web Server

Database Server

Database Server

Database Server

Database Server

Memcache

Memcache

Memcache

Memcache

Memcache
Cloud Computing
Cloud 2006 - Infrastructure as a Service (IaaS)

- Amazon Elastic Compute Cloud (EC2)
  - Web Interface to specify Virtual Machines connected to a virtual network

- Amazon Simple Storage Service (S3)
  - Reliable Data Blob Storage Service

- Usage-based pricing
  - Huge difference: 1000 machines for one min same as one machine for 1000 minutes
  - Matches needs of scale-out architectures

- Interesting business synergy
  - Amazon needed machines for holiday crunch but were available at other times
Early Cloud Computing Users

- **Startup Web App companies**
  - Key enabler: Computing costs start at $0 and scale with additional users
  - Hope: Zero -> billion users

- **Reliable Storage**
  - Three-way geographic replication of data is nice but hard to do on your own

- **Moving a corporate data center to cloud less clear win**
  - Lift and Shift possible but pricing isn't a beneficial as for scale-out users

- **Lots of trust concerns**
  - Someone else has access to our information, what happens if Amazon gets busy, etc.
Cloud computing drivers

● Look for features that both help users and help provider efficiency
  ○ For users: Make easier to program, offload more of the work on infrastructure, cost less
  ○ For provider: Allow better packing onto available compute resources
  ○ Example: Spot marketing pricing - lower prices and better scheduling

● Providers looking to offer higher level functionality
  ○ Provider can offer more value in exchange for more payments
  ○ Example: Provide more of infrastructure and services
Cloud Computing: Platform as a Service (PaaS)

For scale-out services:

- Cloud infrastructure for starting and stop workers based on load demand
- **Containers** (Apps and their dependencies)
  - Easier to specify and schedule processes than virtual machine
- **Serverless** (Functions executed on events)
  - Easier to specify and schedule function execution than containers

Bulk data scale-out operations:

- Map/Reduce, Spark, analytics engines, etc.
Cloud: Software and a Service (SaaS)

- Provider can offer complete applications (e.g. Google docs, Microsoft Office)
- Good way from the provider to get scale and expertise
- Can cause conflicts (e.g. retail store on Amazon?)
Fintech on Cloud
Cloud: Fintech - Deliberate Control

- Batch processing of big data is a sweet spot for the cloud
  - Cloud scale-out support useful for this
  - One issue: Getting data into the cloud

- Cloud versus on-prem for fintech deliberate control
  - Trust: Cloud is a trillion dollar business, bigger than a bank
    - Cloud Provider has lots to lose if something goes wrong
  - Expertise: Cloud hires and practices better IT than banks
    - Example: Security, Machine learning, etc.

- Banks are in the banking business not IT business
  - Unless bank thinks it can get a competitive advantage with IT, it should outsource
Cloud: Fintech - Reflex Control

● Currently co-located in exchanges data centers (low latency gateway access)
  ○ Need exchange to move to cloud to have reflex control in cloud

● Exchanges might like to move to the cloud
  ○ Entire ecosystem: Exchange + Reflex Control + Deliberate control
  ○ Use exchanges for more things (e.g. buying Pizza?)
  ○ Data generation (stock ticker) and consumption within a cloud

● Challenge: Low latency & real time not available on cloud
  ○ Virtualization & sharing overheads of cloud doesn't permit it
  ○ Lift and shift approach won't work
Possible exchange in Cloud options

- Convince a cloud provider it would be worthwhile to offer a low latency, low jitter, real-time cloud for fintech exchanges
  - Possible, if ecosystem is large enough business

- Note: need fairness not necessarily low latency/jitter, real-time
  - Fair in exchanges means whomever is first wins

- Possible "fairness" with accurate clocks....
Timestamp transactions at gateways to establish precise order of arrival

Resequence transactions in a "reordering buffer" before execution

Sequence point

Matcher

Core matching

Sequencer
Switch
Timestamp

... release them simultaneously at (geographically) different locations

Figure from www.eurexchange.com
Thanks - Questions?