Cloud overview
"The illiterate of the 21st century will not be those who cannot read and write, but those who cannot learn, unlearn, and relearn."
- Alvin Toffler, "Future Shock" (1970)

"We're getting better and better at finding the folks with the right attitude and the right history of learning... the world of technology changes so rapidly, that any amount of real experience and depth in a given area has value for about a decade."
- A recruiter (for a company with a 20% policy)
Motivation

- Majority of jobs in CS in the next decade will involve the cloud
  - Companies moving to it
  - Software and services being migrated onto it
  - New services being built on top of it
- Abstractions quickly changing
  - Learn the right level
  - Know what's already been done to focus on what adds value
Have an eye for the future...

- Will network administration jobs obsoleted by software-defined networking?
- Will virtualization software be obsoleted by infrastructure-as-a-service?
- Will database administration jobs obsoleted by database-as-a-service?
- Will IT and Ops jobs be obsoleted by platform-as-a-service (NoOps)?
- Will cloud administration tools be obsoleted by orchestration (Kubernetes)?
Abstractions in communications

- Circa 1980s: program packets directly (IP)
- Circa 1990s: program via sockets as abstractions over TCP and UDP packets (BSD)
- Circa 2000s: program via URLs as abstraction over HTTP/TLS over sockets and packets
Abstractions in software engineering

- Circa 1980s: Single program and machine/OS (C/Asm)
- Circa 1990s: Client-server apps, (Browsers/Servers)
- Circa 2000s: Collections of machines and networks performing complex tasks (Distributed clusters)
- Now: Cloud, multi-cloud applications
Abstractions in systems deployment

- 1990s
  - Purchase your own hardware/software
  - Purchase network capacity from ISP
  - Build-out physical space to host
  - Hire IT and operations staff to manage and deploy
  - Pay for everything, even if not being used
Abstractions in systems deployment

• 2000s
  • Shared data-center
  • Rent rackspace and host servers at a data center co-located with others
  • Same as before, but purchase physical space and network capacity at data center
    • Economies of scale at the hosting provider
  • But...
    • Still requires companies to purchase servers and install software
    • Still requires companies to have their own IT and operations staff
    • e.g. each company has some poor soul carrying a pager 24/7
Abstractions in systems deployment

- Cloud computing
  - Computing resources as a service (like electricity)
Cloud computing advantages

- Can go "assetless"
  - No hardware to purchase, no physical space to purchase
- Can have minimal IT/Ops personnel (comparatively)
  - Simplified, automated IT management by provider
  - Operations handled by provider
  - Exponential growth in machines and network without an exponential growth in employees
    - Requires increasing automation of everything in management
Cloud computing advantages

• Ability to pay only for what you use
  • At per-second granularity for many cloud providers
  • Small up-front investment and low ongoing costs
• Scale up and down on-demand
• Economies of scale over machines, data-centers, and networks
  • Leverage statistical multiplexing gain between companies and their users to reduce overall cost
  • Similar to packet-switched vs. circuit-switched networks
Cloud computing advantages

- Flexibility to go from DevOps (managed by client) to NoOps (managed by cloud provider)
  - Transition being made over the next decade
  - Where job opportunities are
- Value-added services available
  - e.g. ML APIs (Vision, Video Intelligence, Natural Language Processing, Speech)
  - Data warehousing
- Raises the levels of abstraction in systems development and deployment
How?

- **Enabling technology**
  - Full-featured web browsers
  - Fast and inexpensive servers
  - High speed Internet access
  - Large-scale distributed storage and file systems
- **Virtualization**
  - Software (Xen, VMWare, VirtualBox, KVM, HyperV)
  - Hardware (Intel VT-x, AMD SVM)
  - Networks (SDN)
- **Open standards and software**
  - Networks (OpenFlow)
  - OS (Linux, Docker)
  - Applications (nginx, apache2, python, MySQL, Hadoop, Kubernetes etc.)
But, some new problems..

- Security
  - Lower visibility into running services
  - Cross-VM attacks (Spectre, Row-hammer)
  - Intractably complex policies
- Privacy
  - Regulations governing data (FERPA, government surveillance)
- Vendor lock-in
- Migration complexity
Case study: Snapchat (2011)

- Small startup with no data centers, no operations team
- Two developers with a very simple app

Hence, Facebook had to make Poke. Facebook created the app in just 12 days, reportedly after Snapchat turned down Facebook’s attempts to acquire the firm. Poke is ridiculously similar to Snapchat, a feature-for-feature copy that would make Xerox blush. Imitation isn’t unusual for Facebook; as I argued last.
Case study: Snapchat

- By 2013, 10s of millions of users, hundreds of millions of messages
- 2017
  - > 170 million daily users
  - 3 billion photos and videos per day
  - How?

Case study: Snapchat

This is what Snap is paying Google $2 billion for

The social media company has a huge contract with Google Cloud Platform. But what does that mean?

BY TESS TOWNSEND  |  MAR 1, 2017, 5:29PM EST

How Snapchat made a leap of faith by building atop Google cloud services

Stacey Higginbotham  May 7, 2013 - 10:28 AM CDT

Building out the infrastructure for Snapchat was an act of faith, according to co-founder and CTO Bobby Murphy. The company, which apparently was so easy to build that a Facebook engineer took two weeks to mock up a similar service, operates on Google’s (s goog) App Engine. That’s a notable choice in a field of startups that have chosen the more popular cloud services provided by Amazon Web Services (s amzn).
Case study: Snapchat

- App Engine (Platform-as-a-Service)
  - Allows startup company focus on core competency (the app itself), not on the data center or infrastructure
  - Allows fast deployment of new versions
  - Allows app to leverage reliability of Google infrastructure
  - Allows fast scale up to massive client base
Scaling approaches in the cloud

- Vertical vs. Horizontal
- Vertical scaling
  - “Scaling Up”
  - Upgrade machine type
But, beyond 10000 req/sec?
What about downtime during failures?
Scaling approaches

- Horizontal scaling
  - “Scale out”
  - Replicate and load balance
    - As seen in CDN lab…

Create a new instance group

- Region: us-west1
- Specify port name mapping (Optional)
- Instance template: website-template
- Autoscaling: On
- Autoscale based on: HTTP load balancing usage
- Target load balancing usage: 80%
- Minimum number of instances: 1
- Maximum number of instances: 10
Horizontal scaling

- Automatically add more servers to meet demand
  - Elastic, auto-scaling, or managed instances
  - Load balancer distributes based on policy
    - Round-robin, server load, least connections, URL
Deployment Models

- Public Clouds
  - Hosted, operated and managed by third party vendor
  - Security and day to day management by the vendor
- Private Clouds
  - Networks, infrastructures, data centers owned by the organization
- Hybrid Clouds
  - Sensitive applications in a private cloud and non sensitive applications in a public cloud
- Why?
  - Regulatory issues
  - Bandwidth issues (e.g. lack of nearby GCP)
  - Location of data (China)
  - Sunk costs in data centers