Distributed & Cloud Computing

SE 4230
Jay Urbain, Ph.D.

Credits:
Michael Ambrust, et. al., *Above the Clouds: A Berkley View of Cloud Computing*.
Hamilton, J. Cost of Power in Large-Scale Data Centers. November, 2008.
http://www.eia/doe.gov/neic/rankings/stateelectricityprice.htm
Gray, J. Distributed Computing Economics. ACM Queue 6, 3 (2008), 8-17.
http://aws.amazon.com/what-is-cloud-computing/

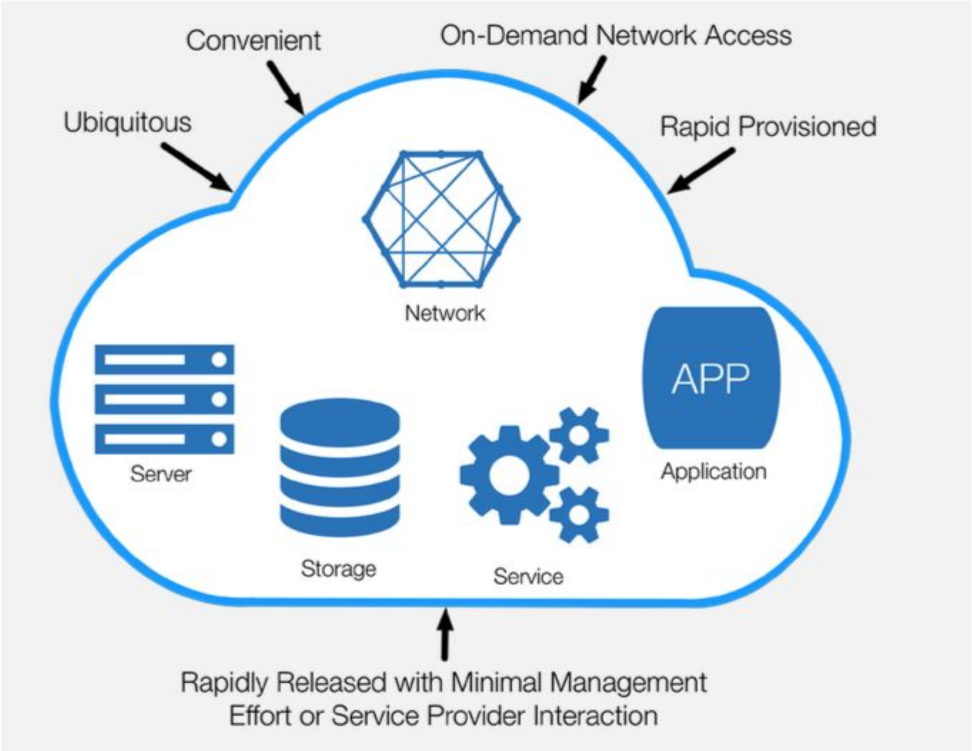

LET'S IMPLEMENT CLOUD COMPUTING SO I HAVE SOMETHING TO TALK ABOUT AT THE EXECUTIVE MEETING.
Dilbert.com DilbertCartoonist@gmail.com
TELL THEM WE'RE EVALUATING IT. THAT WAY NEITHER OF US NEEDS TO DO ANY REAL WORK.
11-18-09 ©2009 Scott Adams, Inc./Dist. by UFS, Inc.
I LIKE IT WHEN YOU DO REAL WORK.
SORRY. I THOUGHT YOU WERE LEADING BY EXAMPLE.

Cloud Computing

Cloud Computing – what are we actually referring to?

Cloud Computing

- Cloud Computing refers to:
  1. The *applications delivered as services* over the Internet.
  2. The *hardware, network, and systems software* in the datacenters that provide those services.



Back to the Future

"Computing may someday be organized as a public utility, just as the telephone system is organized as a public utility."

(John McCarthy, 1961)

Why has this been happening?

Cloud Computing

Cloud Computing –

- Long-held dream of *computing as a utility*
- *Transforming IT industry*
- Make software more attractive as a *service*
- Do you have an innovative idea?

  Need not be concerned about:
  - *Large capital outlays* in hardware to deploy your service or the human expense to operate it.
  - *Overprovisioning* for a service whose popularity does not meet their predictions, wasting costly resources.
  - *Underprovisioning* for one that becomes wildly popular, thus missing potential customers, revenue, and first mover opportunity.

Cloud Service Models

- Software as a Service (SaaS)
  - End user applications - completed product that is run and managed by the service provider.
  - Focus is how you use App, not how its managed.
  - SalesForce.com, Netflix, Snowflake
- Platform as a Service (PaaS)
  - Deployment and management of your applications.
  - Tools and services for deploying customer-created applications to a cloud
  - Google Cloud Platform, AWS Management, Azure, Digital Ocean
- Infrastructure as a Service (IaaS)
  - Basic building blocks – similar to most existing IT resources.
  - Networking, computers (virtual or dedicated), and storage.
  - Capacity and other fundamental computing resources, virtualization.
  - EC2, S3

Cloud Deployment Models

Public:

- Application is fully deployed in the cloud.

Hybrid:

- Deployment between the cloud and existing on-premises infrastructure.

Private/On-premises:

- Deploying resources on-premises using virtualization and resource management tools.
- "Private cloud."

Cloud Computing = SaaS + PaaS + Iaas

Utility Computing= PaaS + Iaas

Cloud Computing – Data Center

Hardware perspective:

- *Illusion of infinite computing resources available on demand.*
  - Eliminates need for Cloud Computing users to plan far ahead for provisioning.
- *Elimination of an up-front commitment by Cloud users.*
  - Allows companies to start small and increase HW only when there is an increase in need
- *The ability to pay for use of computing resources on a short-term basis as needed.*
  - Award conservation by also releasing as needed.

SaaS

Advantages of *SaaS* for developers and end-users:

- Service providers enjoy greatly simplified software installation and maintenance, and centralized control over versioning.
  - *Enhances potential for rapid innovation.*
- End users can access the service “anytime, anywhere”, collaborate more easily, store data safely.
- Seamlessly maintain system qualities.
- Note:
  - Cloud computing does not change SaaS, other than allowing you to deploy services without having to provision a datacenter!

Cloud Computing Provider

Critical characteristics for utility computing:

- Illusion of infinite computing
- Elimination of upfront commitment
- Ability to pay for use

. . . all critical to the success of Cloud Computing

Failed example:

- Past efforts at utility computing without all 3 characteristics have failed. E.g. Intel Computing Services 2000-2001.
- Successful example:

Instance name	On-Demand hourly rate	vCPU	Memory	Storage	Network performance
t3a.nano	$0.0047	2	0.5 GiB	EBS Only	Up to 5 Gigabit
t3a.micro	$0.0094	2	1 GiB	EBS Only	Up to 5 Gigabit
t3a.small	$0.0188	2	2 GiB	EBS Only	Up to 5 Gigabit
t3a.medium	$0.0376	2	4 GiB	EBS Only	Up to 5 Gigabit
t3a.large	$0.0752	2	8 GiB	EBS Only	Up to 5 Gigabit
t3a.xlarge	$0.1504	4	16 GiB	EBS Only	Up to 5 Gigabit
t3a.2xlarge	$0.3008	8	32 GiB	EBS Only	Up to 5 Gigabit

Cloud Computing Provider – Business Opportunity

- Attraction to Cloud Computing for SaaS providers is clear.

- *But why would you want to be a Cloud Computing provider?*
  - Sounds like a commodity business.
  - How to realize *commodities of scale* for building, provisioning, and launching billions of dollar investment in data centers?

Provider Business Opportunity

- Why would you want to be a Cloud Computing provider?
  - Previous growth in Web services compelled Amazon, Google, Microsoft, Salesforce.com, and others to already build out cloud computing infrastructure.
  - Also, had to develop scalable software infrastructure: MapReduce, Google File System, Big Table, and Dynamo.

*Note: Existing investments in large datacenters and large-scale software infrastructure is necessary, but not sufficient conditions to become Cloud Computing vendor – **need business objective!***

Factors Influencing Cloud Computing Providers

1. Make a lot of money
   – Large data centers (tens of thousands of computers) can purchase hardware, network bandwidth, and power for 1/5 to 1/7 the prices offered to a medium sized datacenter (hundreds or thousands of computers).
2. Leverage existing investment
   – Adding CC services on top of existing infrastructures provides a new revenue stream at low incremental cost.
3. Defend a franchise
   – Vendors with established franchises, e.g., Microsoft Azure for migrating desktop apps to cloud.
4. Attack an incumbent
   – Establish beachhead in CC space before dominant provider emerges, e.g., AWS, GCP
5. Leverage customer relationships
   – IBM Global Services, Oracle Database Systems
6. Become a platform
   – Plug-in applications, e.g., algorithmic trading.

Economies of Scale

Economies of scale (2006)*

Technology	Cost in Medium-sized DC	Cost in Very Large DC	Ratio
Network	$95 per Mbit/sec/month	$13 per Mbit/sec/month	7.1
Storage	$2.20 per GByte / month	$0.40 per GByte / month	5.7
Administration	$\approx$140 Servers / Administrator	$>$1000 Servers / Administrator	7.1

**Hamilton, J. Cost of Power in Large-Scale Data Centers. November, 2008.*

Note: Old numbers, but I'm assuming the ratios still hold.

Datacenters and Power

Data centers being built in odd places*

Power in the Data Center and its Cost Across the U.S., 2017

https://info.siteselectiongroup.com/blog/power-in-the-data-center-and-its-costs-across-the-united-states

State Rankings based on Industrial Electricity Rates

Rank	State	Average Industrial Electricity Rate (Cents per kWh)
1	Washington	4.68
2	Montana	5.52
3	Oklahoma	5.58
4	Texas	5.70
5	Kentucky	5.73

New Technology Trends and Business Models

- Construction of large sale commodity-computer datacenters is a key enabler.
- New Technology trends are also playing an important role.
- Shift from *high-touch service* to *low touch, low commitment, self-service*.
- Example 1:
  - Web 1.0 accepting credit card payments required a contractual arrangement with a payment processing service like VeriSIgn.
  - With Stipe, PayPal, any individual can accept credit card payments with no contract, no long-term commitment, and only modes pay-as-you-go transaction fees. Level of touch is almost non-existent.

Example 2: Ad revenue with Google AdSense versus DoubleClick

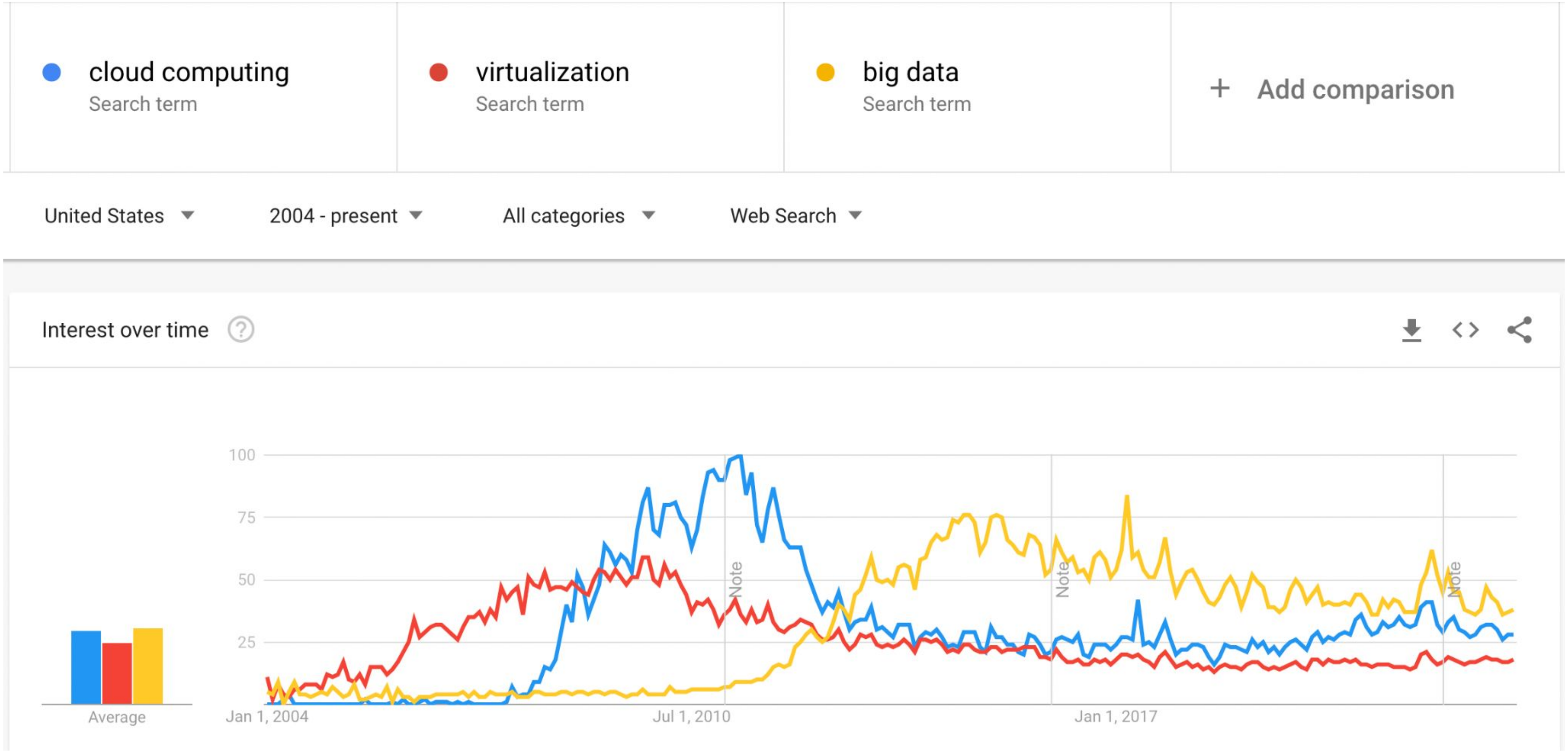
Example 3: Distribute Web content with Amazon CloudFront, CloudFare, Fastly versus Akami.

New Technology Trends

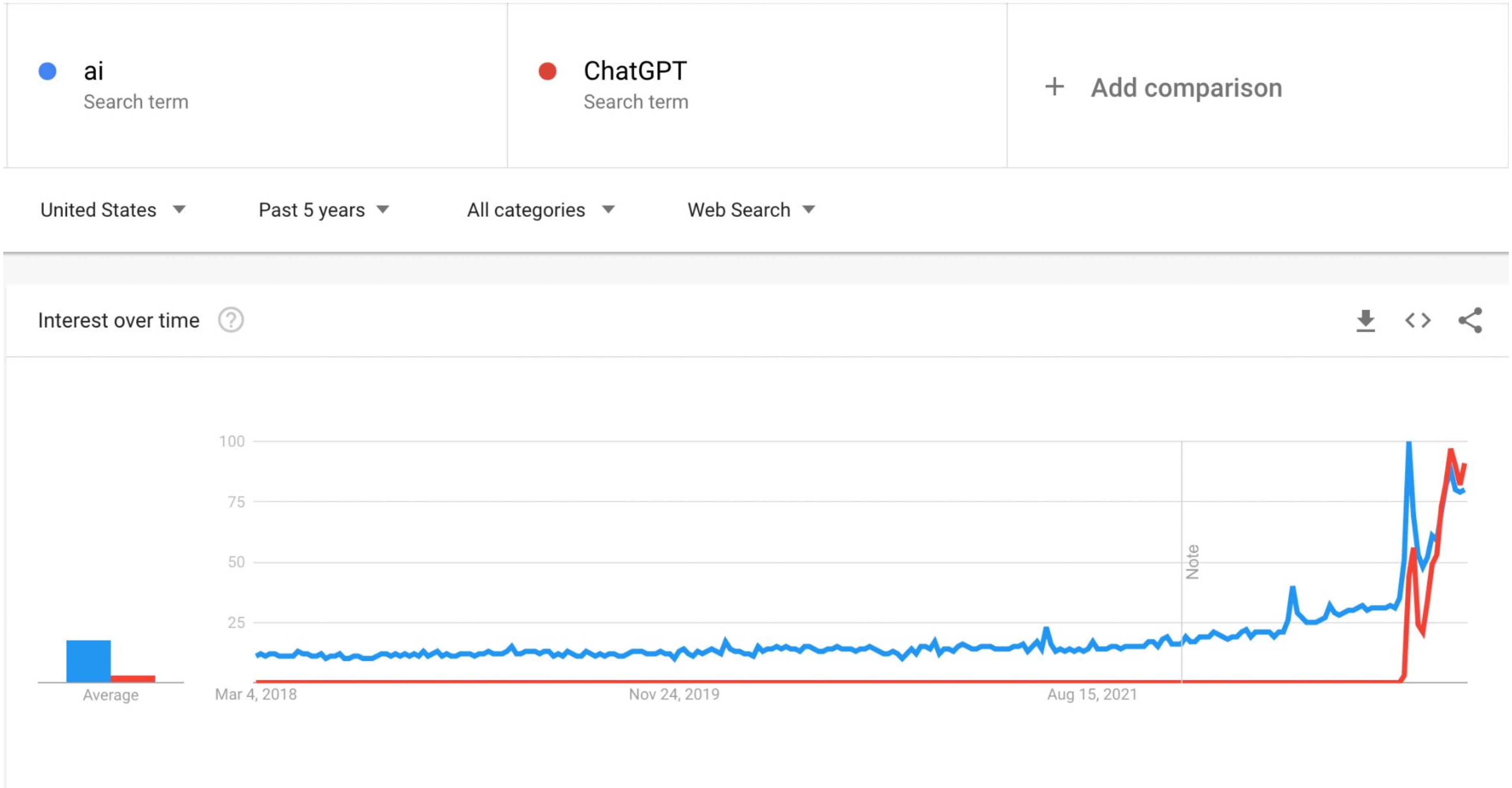
"Economic necessity mandates putting the data near the application, since the cost of wide-area networking has fallen more slowly (and remains relatively higher) than all other IT hardware costs."

Gray, J. Distributed Computing Economics. ACM Queue 6, 3 (2008), 8-17.

Synergy: Cloud computing, Virtualization, & Big Data



Just for Fun


ai
Search term
ChatGPT
Search term
+ Add comparison
United States
Past 5 years
All categories
Web Search
Interest over time
100
75
50
25
Note
Average
Mar 4, 2018
Nov 24, 2019
Aug 15, 2021

Application Opportunities

- Mobile Interactive Computing
  - *Future belongs to services that respond in real time to information provided either by their users or by non-human sensors* – Tim O'Reily.
  - Edge Computing for reduced latency
  - Attracted to cloud for high availability, also rely on large data sets – e.g., mash-ups.

- AI, Large Language Model Services
  - *ChatGPT, Bart, etc.*
  - Training and inference workloads.

- High throughput Data processing Services
  - *CBOE, CME, Citadel, etc.*

Application Opportunities

- Parallel batch processing - Batch processing of analytics jobs that analyze terabytes of data and take hours to finish:
  - Social network analysis
  - Indexing
  - Collaborative filtering
  - Data mining

- The rise of analytics, big data
  - Special case of compute-intensive batch processing is business analytics.
  - Goes beyond transaction processing, understand customer, buying trends, supply chain, ranking, etc.

Application Opportunities

- Extension of compute intensive desktop applications
  - Python stack, R, Matlab, and Mathematica can use CC to perform expensive evaluations.
  - Statistical analysis
  - Machine Learning
  - Data analytics
  - Symbolic math
  - Data visualization
  - Offline rendering
  - Financial quantitative analysis

Application Opportunities

- Earthbound (non-public cloud) applications
  - Fundamental latency limits of getting data into and out of the cloud, or both limit utility computing for some applications.
  - E.g., Realtime financials, stock trading.
  - Need edge computing content delivery solutions: CloudFare, Fastly, etc.

Classes of Utility Computing

Applications require model of computation, storage, and networking.

- *Statistical multiplexing* is necessary to achieve elasticity and the illusion of infinite capacity. Requires resources to be *virtualized* .
- How compute instances are multiplexed and shared can be hidden from the programmer.
- Different *utility computing offerings* can be distinguished based on the *level of abstraction* presented to the programmer and the level of management of resources.

Classes of Utility Computing: Amazon Web Services (AWS)

AWS EC2 (Elastic Cloud Compute):

- EC2 instance looks much like physical hardware, and users can control nearly the entire software stack, from the kernel upwards.
- The API exposed is “thin” – a few dozen calls to request and configure the virtualized hardware.
- No a prior limit on the kinds of applications that can be hosted.
- Low level of virtualization – raw CPU cycles, lock-device storage, IP-level connectivity – allow developers to code whatever they want.

GCP (Compute Engine).

Classes of Utility Computing: Google AppEngine, AWS Elastic Beanstalk

Domain-specific platforms:

- Google AppEngine, AWS Elastic Beanstalk, Digital Ocean droplet.
- Force.com, the SalesForce business software development platform.

AppEngine

- Targeted at traditional web applications, mobile web services.
- Enforces an application structure of clean separation between a stateless computation tier and a stateful storage tier.
- Applications are expected to be request-reply based (ReST), as such they are rationed in how much CPU time they can use in servicing a particular request.
- Impressive automatic scaling and high-availability mechanisms.
- Proprietary MegaStore (based on BigTable) data storage available to AppEngine applications, all rely on these constraints.
- Thus, AppEngine is not as suitable for general-purpose computing.

Force.com

- Designed to support business applications that run against the salesforce.com database, and nothing else.

Classes of Utility Computing: Containerization

- AWS, GCP, Azure and Digital Ocean all have solutions for running containerized apps.

Cloud Computing Economics

- Deciding whether hosting a service in the cloud makes sense over the long term. Considerations:
  - Economic models enabled by CC make tradeoff decisions more fluid, e.g., risk transfer.
  - HW costs continue to decline, but at different rates, e.g., storage versus WAN costs.
  - Expected average and peak utilization.

Elasticity & Shifting of Risk

Elasticity and shifting of risk (under-provisioning & over-provisioning):

- Converting capital expenses to operating expenses, i.e., pay-as-you-go. Facilitates activity based accounting.
- CC resources can be distributed non-uniformly in time: usage based pricing. Example, SnowFlake!
- Absence of up-front capital expense.

Elasticity

- Add/remove resources at a fine grain.
- Lead time of minutes versus weeks.
- Estimates of server utilization in data centers range from 1% to 20%*.

**RANGAN, K. The Cloud Wars: $100+ billion at stake. Tech. rep., Merrill Lynch, May 2008.*

**SIEGELE, L. Let It Rise: A Special Report on Corporate IT. The Economist (October 2008).*

Example: Elasticity

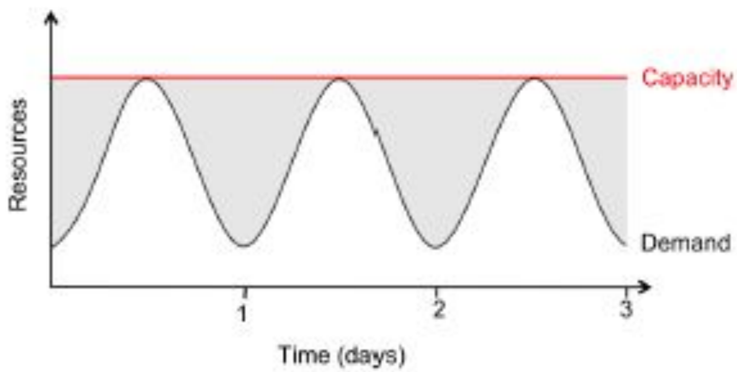
- Assume your service requires *500 servers at noon*, but only *100 servers at midnight*.
- If the *average utilization* over a whole day is *300 servers*, the actual utilization over the whole day is *300*24 = 7200 server-hours.*
- But since you must provision to the peak of 500 servers, you have to *pay for 500*24 = 12000 server-hours*, a *factor of 1.7* more than what is needed.

Example: Elasticity cont.

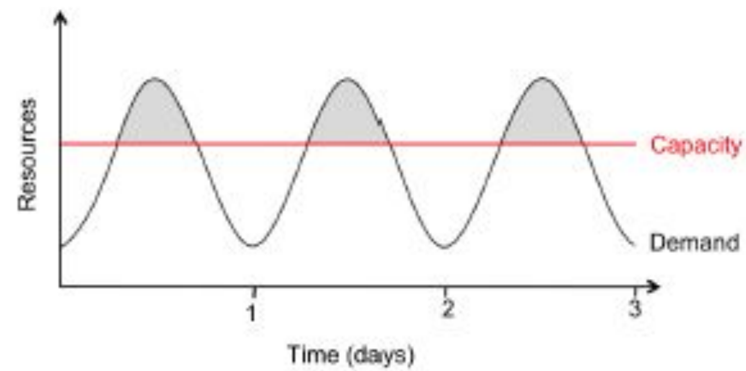
Previous example underestimates the benefit of elasticity.

- Besides daily variation there is seasonal variation.
- Unexpected utilization, episodic events.
- We’re wrong!

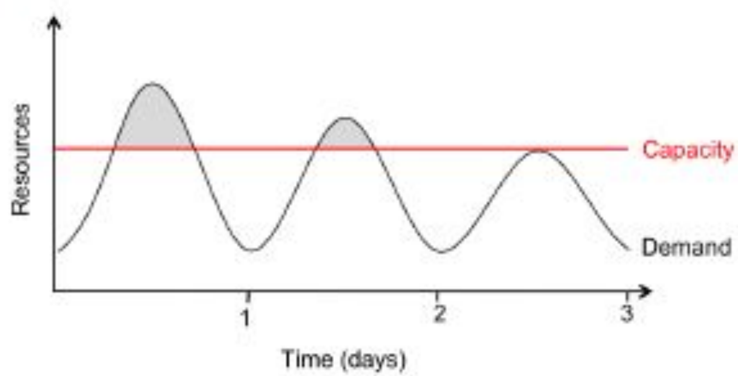
Provisioning



(a) Provisioning for peak load



(b) Underprovisioning 1



(c) Underprovisioning 2

Risk Transfer

Opportunity costs:

- Suppose 10% of users who receive poor service due to under provisioning are “permanently lost” opportunities, i.e. users who would have remained regular visitors with a better experience.
- *Risk of miscalculating work load is shifted onto cloud provider!*

Comparing Costs

Cloud versus fixed-capacity datacenter

- If Utilization = 1.0 (the datacenter equipment is 100% utilized), the two sides of the equation look ~ the same.
- Queuing theory tells us that as utilization approaches 1.0, system response time approaches infinity.
- Revenue generation model.

Fixed cost

$$\mathrm{UserHours}_{\mathrm{cloud}} \times (\mathrm{revenue} - \mathrm{Cost}_{\mathrm{cloud}}) \geq \mathrm{UserHours}_{\mathrm{datacenter}} \times \left(\mathrm{revenue} - \frac{\mathrm{Cost}_{\mathrm{datacenter}}}{\mathrm{Utilization}}\right)$$

Variable

Variable

Move to Cloud? Example

Example:

- Biology lab creates *500 GB of new data* for every wet lab experiment.
- A computer the speed of one EC2 instance takes *2 hours per GB* to process the new data.
- The lab has the equivalent *20 instances* locally, so the time to evaluate the experiment is *500*2/20 or 50 hours*.
- They could process it in a *single hour on 1000 instances* at AWS.
- The cost to process one experiment would be just *1000*$0.10 or $100* in computation and another *500*$0.10 or $50* in network transfer fees.
- So far, so good. They measure the transfer rate from the lab to AWS at *20 Mbits/second*.
- Transfer time:

(500GB * 1000MB/GB * 8bits/Byte * 1sec/20Mbits) * (1 hour/3600 seconds)

> 500*1000*8/20*(1/3600)

[1] 55.55556

- Thus, it takes 50 hours locally vs. *~55 + 1 ! -> 56 hours* on AWS, so they don't move to the cloud.

Number 1 Obstacle: Availability of a Service

Scenario:

- Availability of service.

Tactic:

- Mitigate using multiple networks or service providers.
- Service is actually pretty reliable. ~99+ percentile.

Service and Outage	Duration	Date
S3 outage: authentication service overload leading to unavailability [39]	2 hours	2/15/08
S3 outage: Single bit error leading to gossip protocol blowup. [41]	6-8 hours	7/20/08
AppEngine partial outage: programming error [43]	5 hours	6/17/08
Gmail: site unavailable due to outage in contacts system [29]	1.5 hours	8/11/08

Number 2 Obstacle: Data Lock-In

Scenario:

- Software stacks have improved interoperability among platforms, but the APIs for Cloud Computing itself are still essentially proprietary.

Tactic:

- Standardize the APIs so that a SaaS developer could deploy services and data across multiple Cloud Computing providers.
- Use containers.
- Use 3rd party vendors.

Number 3 Obstacle: Data Confidentiality and Auditability

Scenario:

- Current cloud offerings are essentially public (rather than private) networks, exposing the system to more attacks.
- Requirements for auditability, in the sense of Sarbanes-Oxley and Health and Human Services Health Insurance Portability and Accountability Act (HIPAA) regulations that must be provided for corporate data to be moved to the cloud.
- This can be a real issue, but is going away.

Tactic:

- No fundamental obstacles to making a cloud-computing environment as secure as the vast majority of in-house IT environments.
- Many of the obstacles can be overcome immediately with well understood technologies such as encrypted storage,

Number 4 Obstacle: Data Transfer Bottlenecks

Scenario:

- Applications continue to become more data-intensive.

Tactic:

- This can be a real issue.
- Keep data in cloud.
- Overcome the high cost of Internet transfers is to ship disks.

Number 5 Obstacle: Performance Unpredictability

Scenario:

- Multiple Virtual Machines share CPUs and main memory.

Tactic:

- Cloud provider:
  - Xen VM, VMWare work well in practice.
  - improve architectures and operating systems to efficiently virtualize interrupts and I/O channels.
  - flash “disk” memory will decrease I/O interference.
  - Dedicated hardware
- Cloud user:
  - Increase redundancy, increase nodes.
  - Dedicated instances.

Number 6 Obstacle: Scalable Storage

Scenario:

- Fluctuating demands for storage.

Tactic:

- Complexity of data structures that are directly supported by the storage system (e.g., schema-less blobs vs. column-oriented storage).
- *Open research problem*, is to create a storage system would not only meet these needs but combine them with the cloud advantages of scaling arbitrarily up and down on-demand, as well as meeting programmer expectations in regard to resource management for scalability, data durability, and high availability.
- New offerings in scalable storage.

Number 7 Obstacle: Bugs in Large-Scale Distributed Systems

Scenario:

- Removing errors in these very large scale distributed systems.
- Bugs cannot be reproduced in smaller configurations

Tactic:

- Research more sophisticated debugging tools.
  - Datadog, Dynatrace.
  - Cloud providers.
  - Develop own monitoring infrastructure.
- Develop/debug locally, if possible.

Number 8 Obstacle: Scaling Quickly

Scenario:

- Fluctuating demand.

Tactic:

- Depends on the virtualization level.
- Google AppEngine, and AWS Beanstalk automatically scale in response to load increases and decreases, and users are charged by the cycles used.
- AWS charges by the hour for the number of instances you occupy, even if your machine is idle. Beanstalk is usage.

Number 9 Obstacle: Reputation

Scenario:

- Reputations do not virtualize well.
- One customer's bad behavior can affect the reputation of the cloud as a whole.
- E.g., blacklisting EC2 addresses.

Tactic:

- Competition between cloud vendors.

Number 10 Obstacle: Software Licensing

Scenario:

- Current software licenses commonly restrict the computers on which the software can run.

Tactic:

- Software vendors are figuring out that they need to support CC.
- Use open source!

Conclusions

- Vision of computing as a utility has emerged.
- Cloud providers view the construction of very large data centers at low cost sites using commodity computing, storage, and networking uncovered the possibility of selling those resources on a pay-as-you-go model.
- As Cloud Computing users, we were relieved of dealing with the twin dangers of over-provisioning and under-provisioning our internal data centers.

STOP

networking) and ensures scalability and high availability of the resources.

	Amazon Web Services	Microsoft Azure	Google AppEngine
Computation model (VM)	• x86 Instruction Set Architecture (ISA) via Xen VM • Computation elasticity allows scalability, but developer must build the machinery, or third party VAR such as RightScale must provide it	• Microsoft Common Language Runtime (CLR) VM; common intermediate form executed in managed environment • Machines are provisioned based on declarative descriptions (e.g. which "roles" can be replicated); automatic load balancing	• Predefined application structure and framework; programmer-provided "handlers" written in Python, all persistent state stored in MegaStore (outside Python code) • Automatic scaling up and down of computation and storage; network and server failover; all consistent with 3-tier Web app structure
Storage model	• Range of models from block store (EBS) to augmented key/blob store (SimpleDB) • Automatic scaling varies from no scaling or sharing (EBS) to fully automatic (SimpleDB, S3), depending on which model used • Consistency guarantees vary widely depending on which model used • APIs vary from standardized (EBS) to proprietary	• SQL Data Services (restricted view of SQL Server) • Azure storage service	•MegaStore/BigTable
Networking model	• Declarative specification of IP-level topology; internal placement details concealed • Security Groups enable restricting which nodes may communicate • Availability zones provide abstraction of independent network failure • Elastic IP addresses provide persistently routable network name	• Automatic based on programmer's declarative descriptions of app components (roles)	• Fixed topology to accommodate 3-tier Web app structure • Scaling up and down is automatic and programmer-invisible