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/

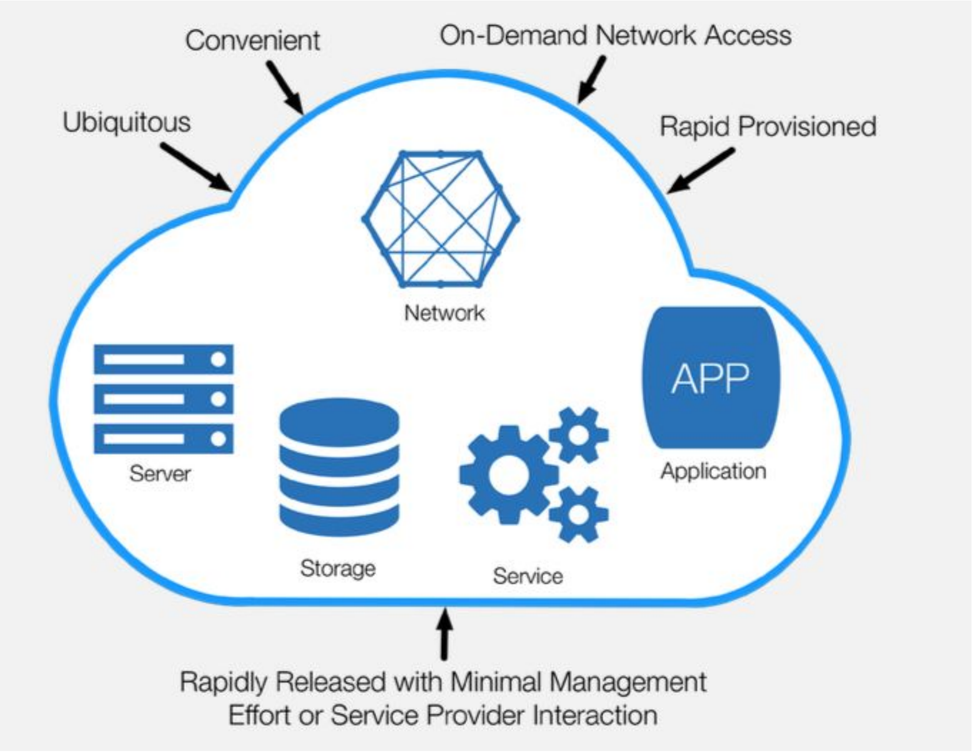

LET'S IMPLEMENT CLOUD COMPUTING SO I HAVE SOMETHING TO TALK ABOUT AT THE EXECUTIVE MEETING.
Dilbert.com DilbertCartoonist@gmail.com
TELL THEM WE'RE EVALUATING IT. THAT WAY NEITHER OF US NEEDS TO DO ANY REAL WORK.
11-18-09 © 2009 Scott Adams, Inc./Dist. by UFS, Inc.
I LIKE IT WHEN YOU DO REAL WORK.
SORRY.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



Roy Campbell & Reza Farivar

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

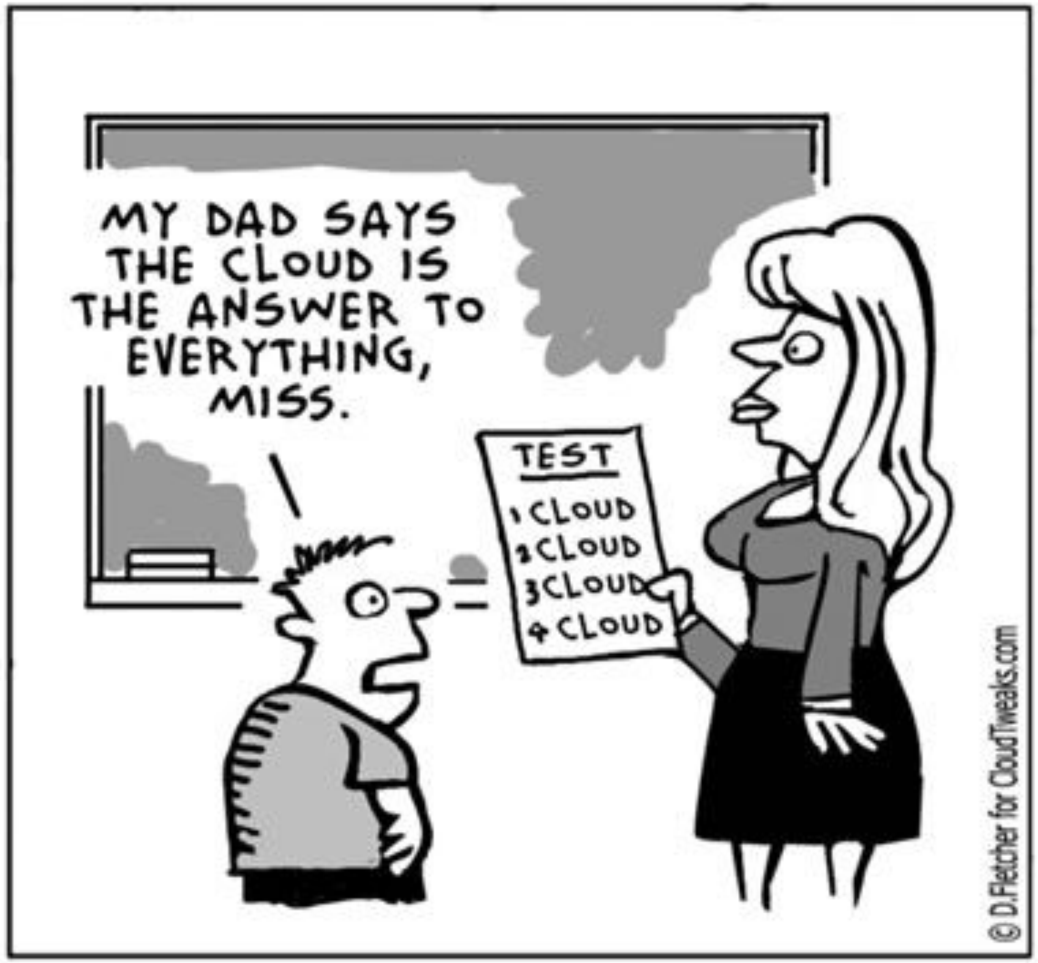
. . . all critical to the success of Cloud Computing

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

START


MY DAD SAYS THE CLOUD IS THE ANSWER TO EVERYTHING, MISS.
TEST
1 CLOUD
2 CLOUD
3 CLOUD
4 CLOUD
© D.Fletcher for CloudTweaks.com

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.

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.
   - Have expertise.
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, Azure.
5. Leverage customer relationships
   - IBM Global Services, Oracle Database Systems
6. Become a platform
   - Plug-in applications, SalesForce Force platform, e.g., algorithmic trading.

Types of Cloud Services

Infrastructure as a Service (IaaS): VMs, disks

Platform as a Service (PaaS): Web, MapReduce

Software as a Service (SaaS): Email, GitHub

Public vs private clouds:

Shared across arbitrary orgs/customers
vs internal to one organization

PaaS Example

AWS Lambda functions-as-a-service

- » Runs functions in a Linux container on events
- » Used for web apps, IoT apps, stream processing, highly parallel MapReduce and video encoding

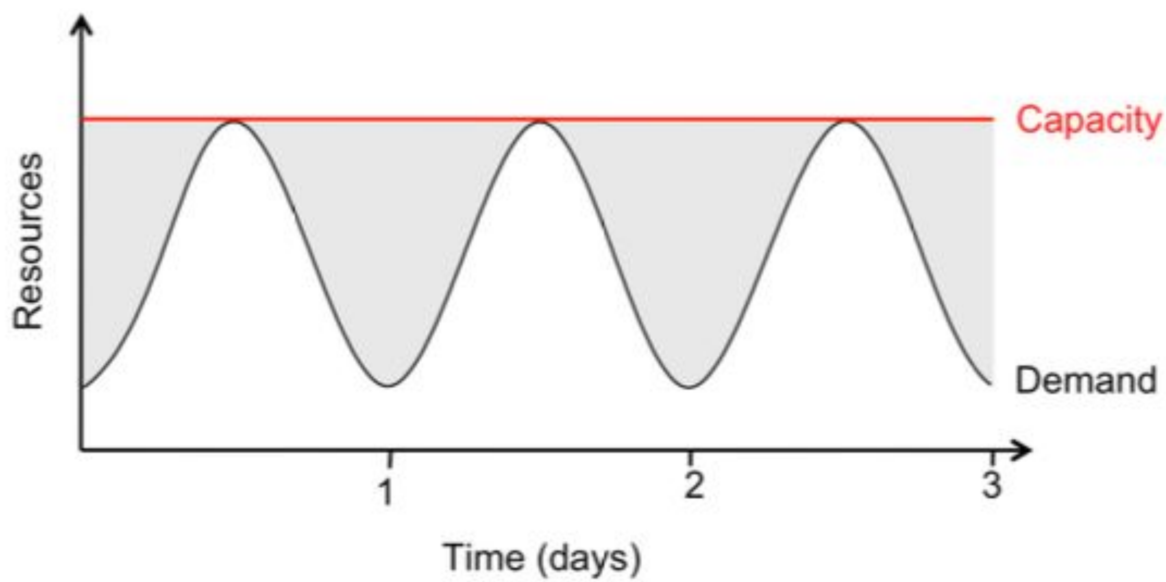
Cloud Economics: For Users

Pay-as-you-go (usage-based) pricing:

» Most services charge per minute, per byte, etc

» No minimum or up-front fee

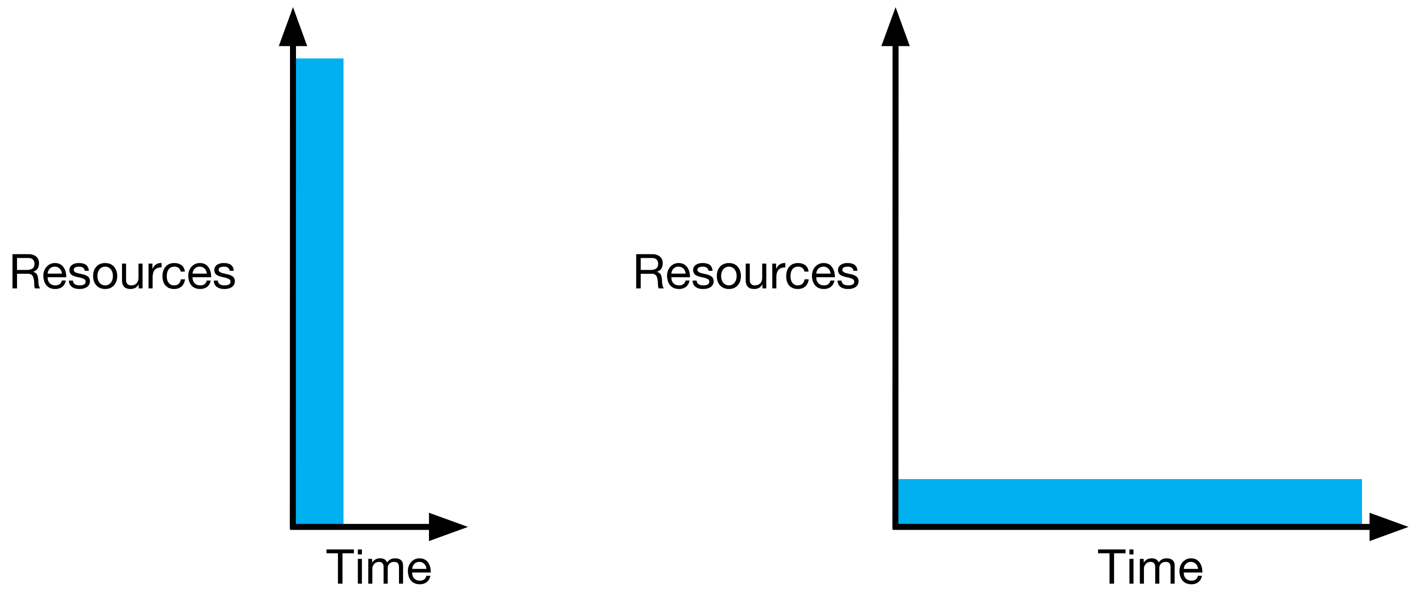
» Helpful when apps have *variable utilization*



Cloud Economics: For Users

Elasticity:

- » Using 1000 servers for 1 hour costs the same as 1 server for 1000 hours
- » Same price to get a result faster!



Cloud Economics: For Providers

Economies of scale:

- Purchasing, powering & managing machines at scale gives lower per-unit costs than customers’
- Tradeoff: fast growth vs efficiency
- Tradeoff: flexibility vs cost

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

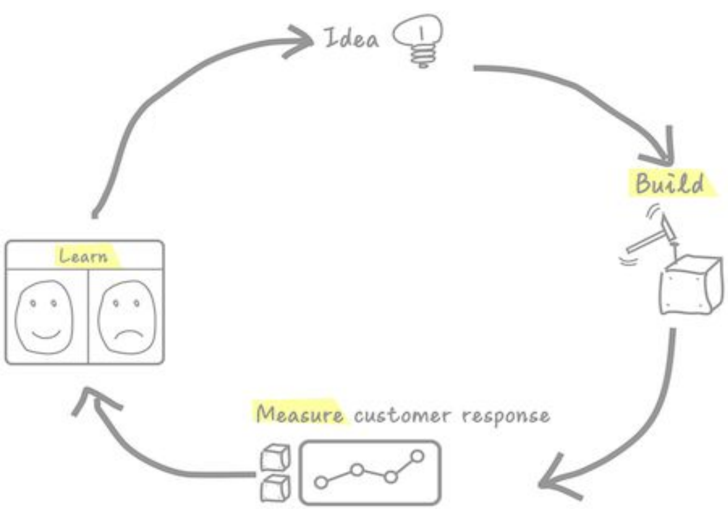
Rank	State	Average Industrial Electricity Rate (Cents per kWh)
1	Washington	4.68
2	Montana	5.52
3	Oklahoma	5.58
4	Texas	5.70
5	Kentucky	5.73

Cloud Economics

Speed of iteration:

» Software as a service means fast time-to-market, updates, and detailed monitoring/feedback

» Compare to speed of iteration with ordinary software distribution



Questions

- Assume you are a cloud provider

  How do you avoid having many of your customers spike at the same time?

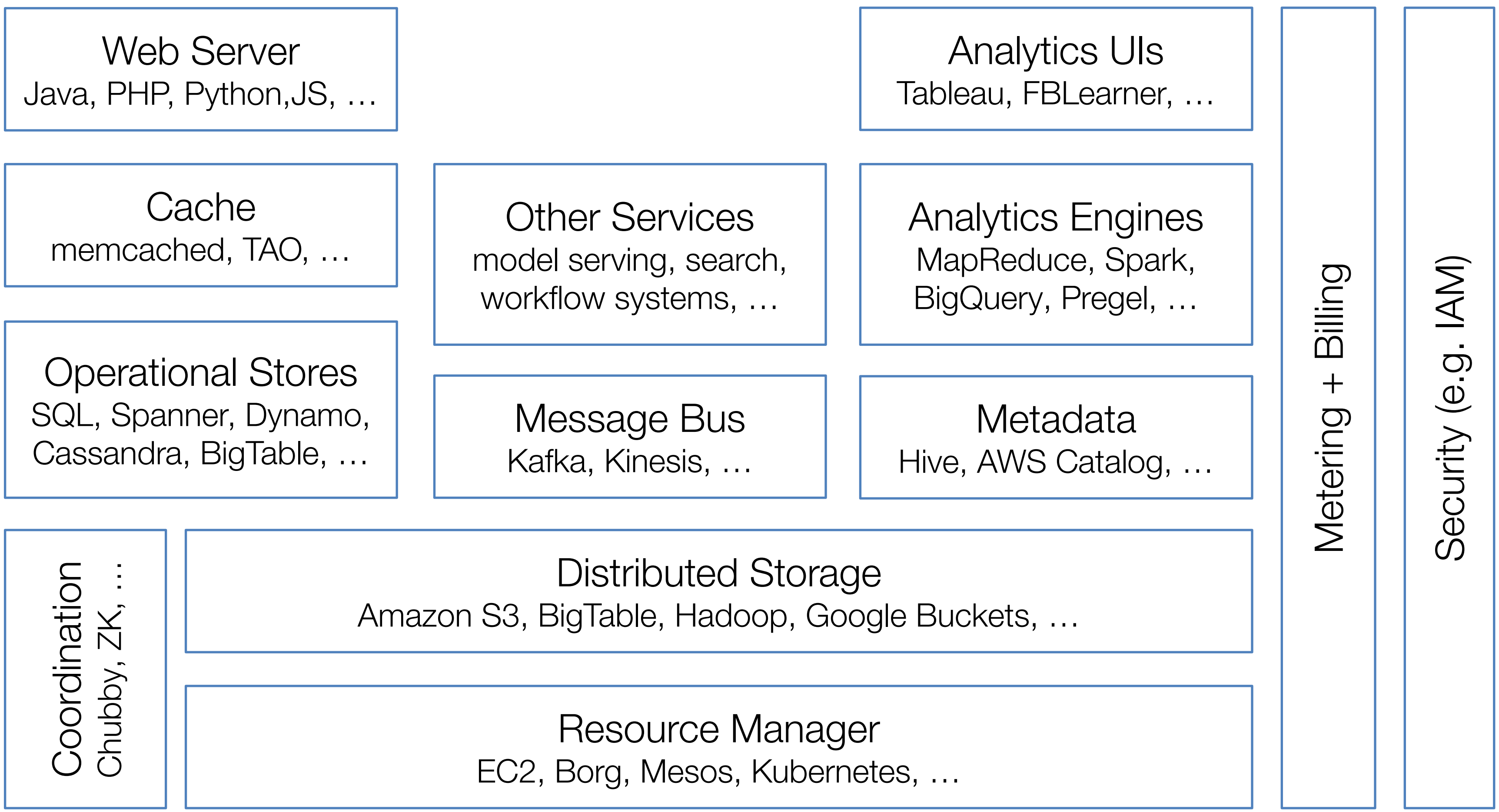
Other Interesting Features

- Spot market for preemptible machines
- Wide geographic access for disaster recovery and speed of access
- Ability to quickly try exotic hardware
- Ability to A/B test anything

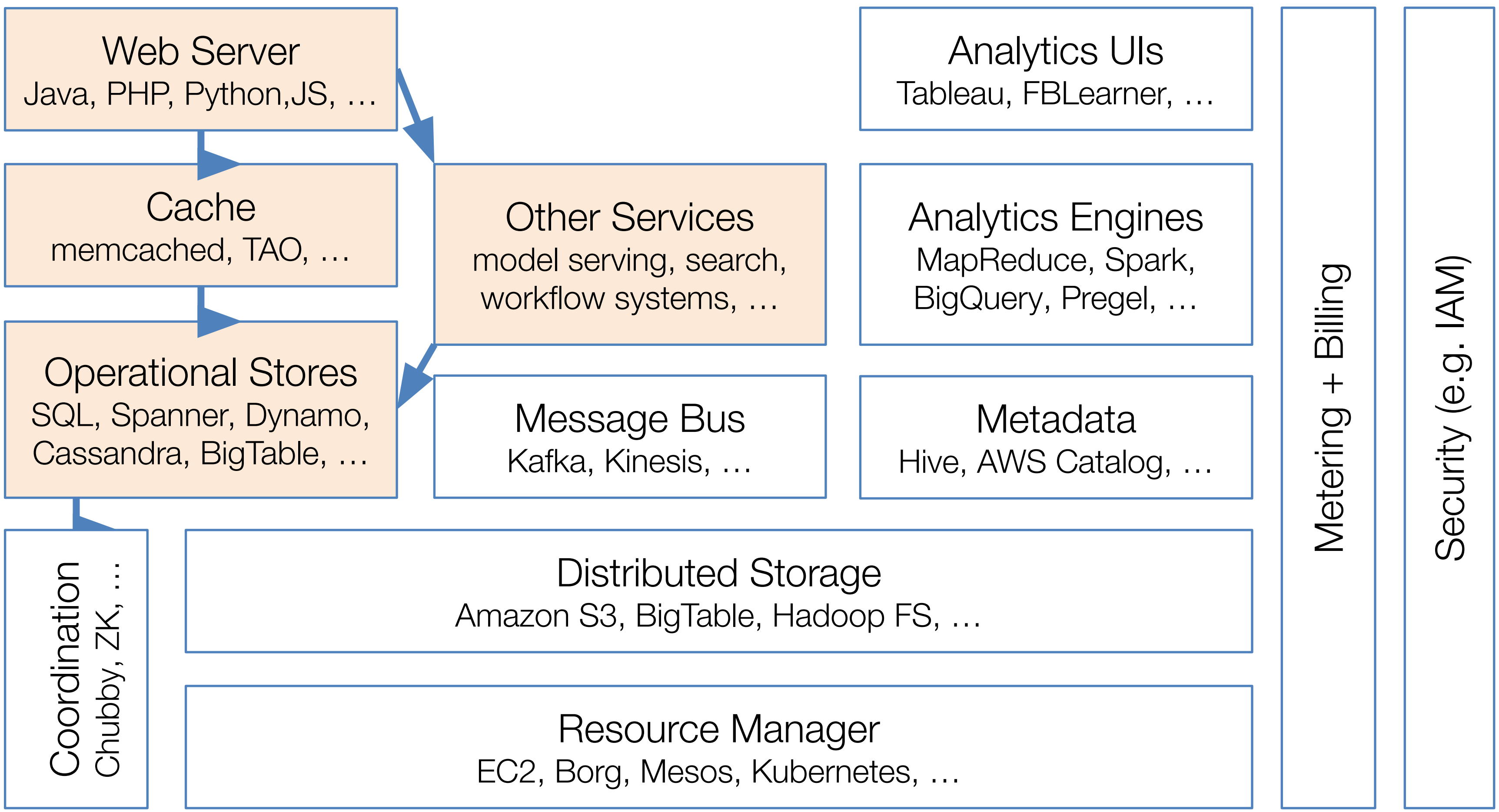
Common Cloud Applications

1. Web and mobile applications
2. Data analytics (MapReduce, SQL, ML, etc)
3. Stream processing
4. Batch computation (HPC, video, etc)
5. Machine learning training and inference workloads.

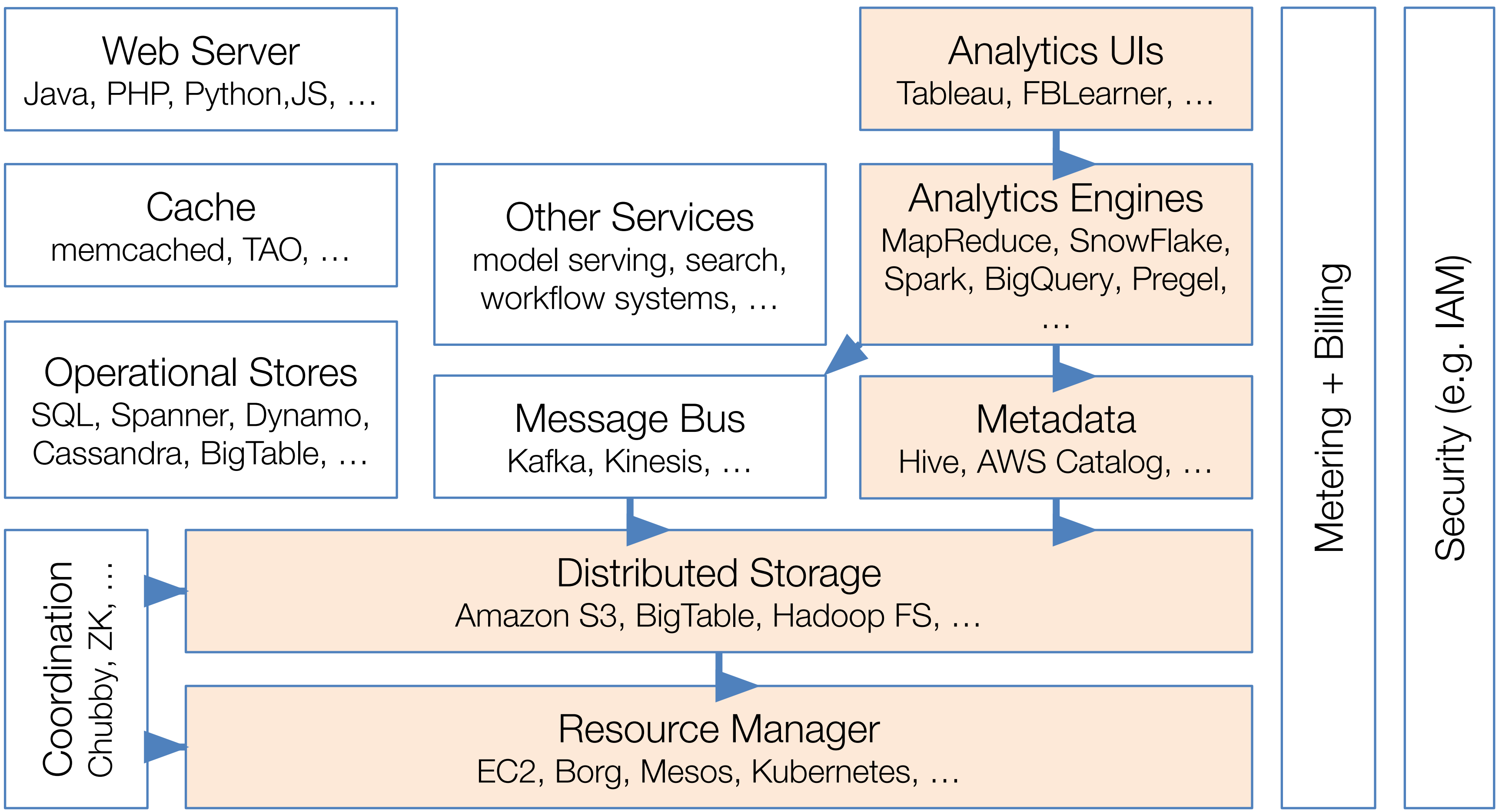
Cloud Software Stack



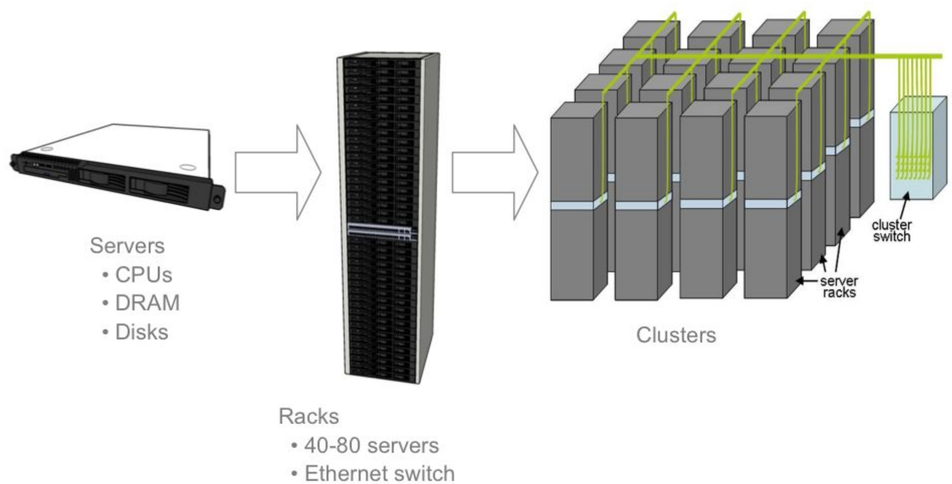
Example: Web Application



Example: Analytics Warehouse



Datacenter Hardware

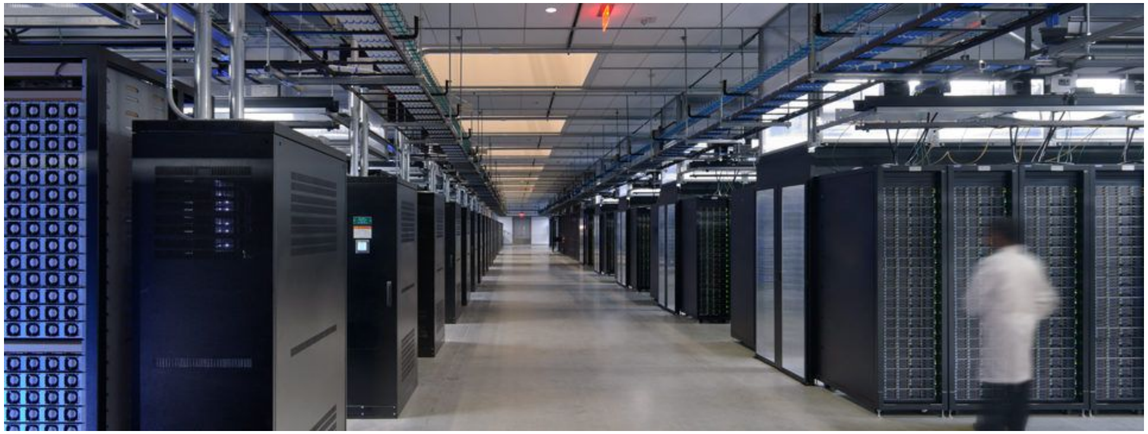


Rows of rack-mounted servers

Datacenter: 50 – 200K servers, 10 – 100MW

Often organized as few and mostly independent clusters

Datacenter Example

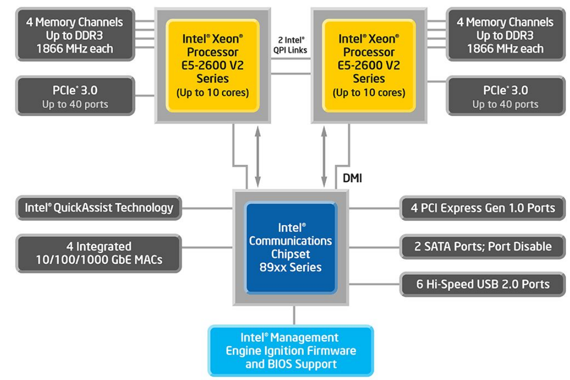


Datacenter HW: Compute

The basics

- Multi-core CPU servers
- 1 & 2 sockets

2-socket server



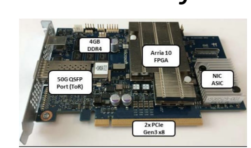
What's new

- GPUs
- FPGAs
- Custom accelerators (AI)

Google TPU^2

Nvidia Tesla

Microsoft Catapult



Hardware Heterogeneity

Standard Systems	I Web	III Database	IV Hadoop	V Haystack	VI Feed
CPU	High 2 x E5-2670	High 2 x E5-2660	High 2 x E5-2660	Low 1 x E5-2660	High 2 x E5-2660
Memory	Low 16GB	High 144GB	Medium 64GB	Med-Hi 96GB	High 144GB
Disk	Low 250GB	High IOPS 3.2 TB Flash	High 15 x 4TB SATA	High 30 x 4TB SATA	Medium 2TB SATA + 1.6TB Flash
Services	Web, Chat	Database	Hadoop	Photos, Video	Multifeed, Search, Ads

[Meta/Facebook server configurations]

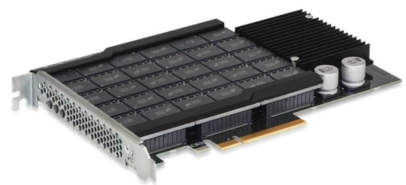
Custom-design servers

- Configurations optimized for major app classes
- Few configurations to allow reuse across many apps
- Roughly constant power budget per volume

Datacenter HW: Storage

JBOD disk array

NVMe Flash



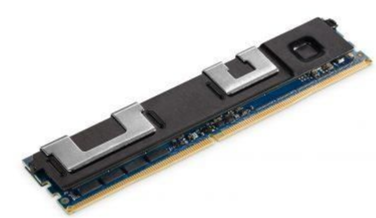
The basics

- Disk trays
- SSD & NVM Flash

What's new

- Non-volatile memories
- New archival storage (e.g., glass)

NVM DIMM



Distributed with compute or NAS systems

- **NAS - Network addressed storage**
- Remote storage access for many use cases (why?)

Datacenter HW: Networking

The basics

- 10, 25, and 40GbE NICs
- 40 to 100GbE switches
- Clos topologies

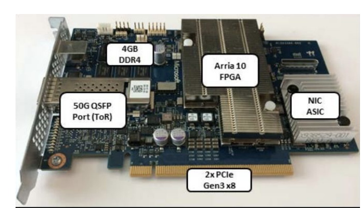
40GbE Switch

Smart NIC

What's new

- Software defined networking
- Smart NICs
- FPGAs

Microsoft Catapult



Useful Latency Numbers

Initial list from Jeff Dean, Google

L1 cache reference	**0.5 ns**
Branch mispredict	5 ns
L3 cache reference	20 ns
Mutex lock/unlock	25 ns
Main memory reference	**100 ns**
Compress 1K bytes with Snappy	3,000 ns
Send 2K bytes over 10Ge	2,000 ns
Read 1 MB sequentially from memory	**100,000 ns**
Read 4KB from NVMe Flash	50,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA → Europe → CA	**150,000,000 ns**

Cloud Computing Economics

- Deciding whether hosting a service in the cloud makes sense over the long term.
  - 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.

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.

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

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

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.

Key Availability Techniques

Technique	Performance	Availability
Replication	✔	✔
Partitioning (sharding)	✔	✔
Load-balancing	✔	
Watchdog timers		✔
Integrity checks		✔
Canaries		✔
Eventual consistency	✔	✔

- Make apps do something reasonable when not all is right
    - Better to give users limited functionality than an error page
- Aggressive load balancing or request dropping
    - Better to satisfy 80% of the users rather than none

The CAP Theorem

In distributed systems, choose 2 out of 3

Consistency

Every read returns data from most recent write

Availability

Every request executes & receives a (non-error) response

Partition-tolerance

The system continues to function when network partitions occur (messages dropped or delayed)

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

Number 1 Obstacle: Availability of a Service

Scenario:

- Availability of service.
- Service is actually pretty reliable. ~99+ percentile.

Tactic:

- Mitigate using multiple networks or service providers.

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.
- Kafka, Confluent.

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!

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