Cloud Computing and the RAD Lab

David Patterson, UC Berkeley
Reliable Adaptive Distributed Systems Lab

(with lots of help from Armando Fox
and a cast of 1000s)

Image: John Curley http://www.flickr.com/photos/jay_que/1834540/

Outline

- What is Cloud Computing?
- Software as a Service / Cloud Computing in Education at UC Berkeley
- UC Berkeley RAD Lab Research Program in Cloud Computing
- Q&A

Clod computing

"Cloud computing is nothing (new)"

"...we've redefined Cloud Computing to include everything that we already do... I don't understand what we would do differently ... other than change the wording of some of our ads."

Larry Ellison, CEO, Oracle (Wall Street Journal, Sept. 26, 2008)

Above the Clouds: A Berkeley View of Cloud Computing

abovetheclouds.cs.berkeley.edu

- 2/09 White paper by RAD Lab PI's and students
  - Shorter version: "A View of Cloud Computing," *Communications of the ACM*, April 2010
  - Clarify terminology around Cloud Computing
  - Quantify comparison with conventional computing
  - Identify Cloud Computing challenges & opportunities
  - 50,000 downloads of paper!
- Why can we offer new perspective?
  - Strong engagement with industry
  - Using cloud computing in research, teaching since 2008
- Goal: stimulate discussion on *what's really new*

Utility Computing Arrives

- Amazon Elastic Compute Cloud (EC2)
- “Compute unit” rental: $0.08-0.64/hr.
  - 1 CU ≈ 1.0-1.2 GHz 2007 AMD Opteron/Xeon core

“Instances”	Platform	Cores	Memory	Disk
Small - $0.08 / hr	32-bit	1	1.7 GB	160 GB
Large - $0.32 / hr	64-bit	4	7.5 GB	850 GB – 2 spindles
XLarge - $0.64 / hr	64-bit	8	15.0 GB	1690 GB – 3 spindles

- No up-front cost, no contract, no minimum
- Billing rounded to nearest hour; pay-as-you-go storage also available
- A new paradigm (!) for deploying services?

What is it? What's new?

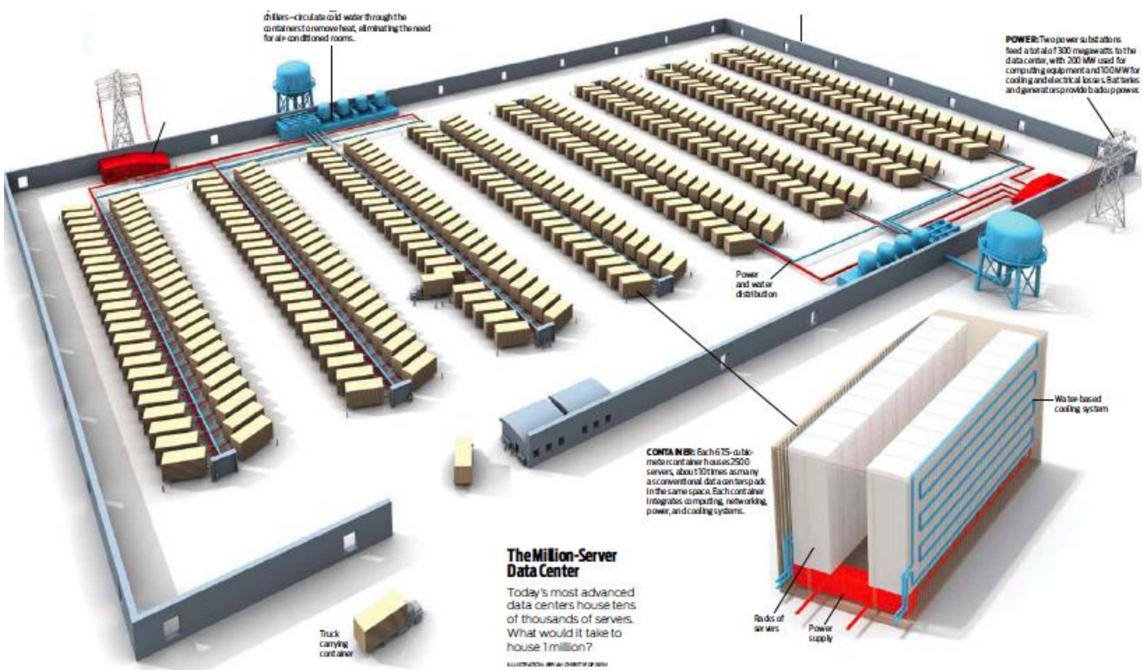
- Old idea: Software as a Service (SaaS)
  - Basic idea predates MULTICS (timesharing in 1960s)
  - Software hosted in the infrastructure vs. installed on local servers or desktops; dumb (but brawny) terminals
  - Recently: "[HW, Infrastructure, Platform] as a service" ?? HaaS, IaaS, PaaS poorly defined, so we avoid
- **New:** pay-as-you-go *utility computing*
  - Illusion of infinite resources on demand
  - Fine-grained billing: release == don't pay
  - Earlier examples: Sun, Intel Computing Services—longer commitment, more $$$/hour, no storage
  - *Public (utility)* vs. *private* clouds

Why Now (not then)?

- "The Web Space Race": Build-out of extremely large datacenters (10,000's of ***commodity*** PCs)
  - Build-out driven by growth in demand (more users)
  - => Infrastructure software: e.g., Google File System
  - => Operational expertise: failover, DDoS, firewalls...
  - Discovered economy of scale: 5-7x cheaper than provisioning a medium-sized (100's machines) facility
- More pervasive broadband Internet
- Commoditization of HW & SW
  - Fast Virtualization
  - Standardized software stacks

Datacenter is the new Server

Utility computing: enabling innovation in new services without first building & capitalizing a large company.

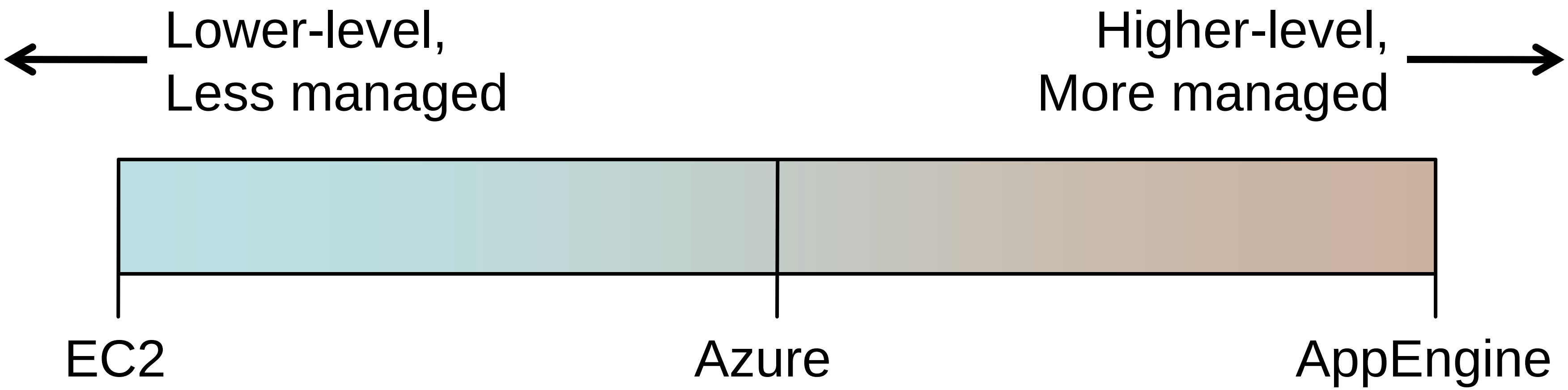


The Million Server Datacenter

- 24000 sq. m housing 400 containers
  - Each container contains 2500 servers
  - Integrated computing, networking, power, cooling systems
- 300 MW supplied from two power substations situated on opposite sides of the datacenter
- Dual water-based cooling systems circulate cold water to containers, eliminating need for air conditioned rooms

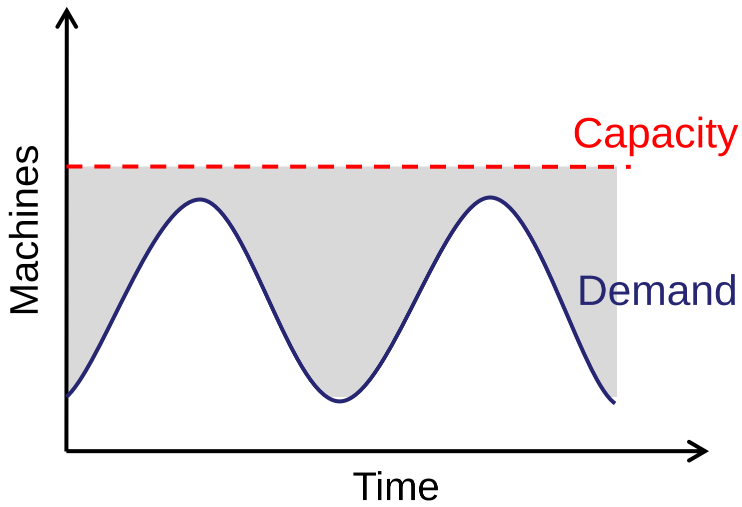
Classifying Clouds

- Instruction Set VM (Amazon EC2)
- Managed runtime VM (Microsoft Azure)
- Framework VM (Google AppEngine)
- *Tradeoff: flexibility/portability vs. "built in" functionality*

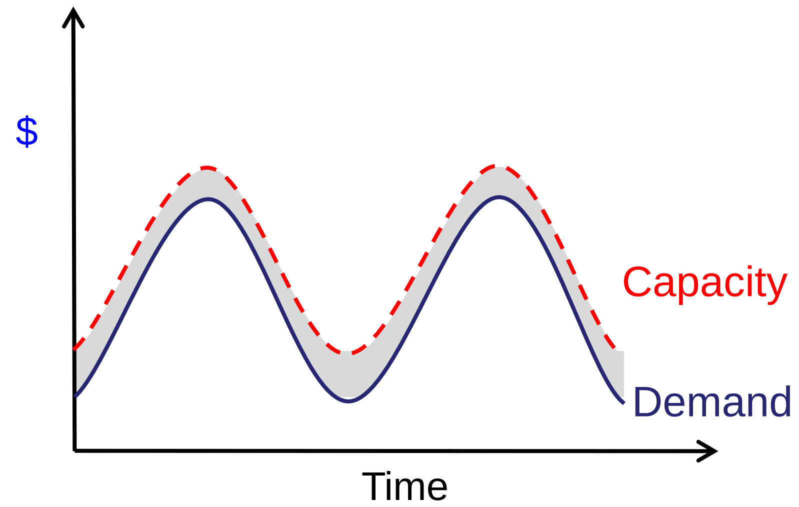


Cloud Economics 101

- Cloud Computing User: Static provisioning for peak - wasteful, but necessary for SLA



"Statically provisioned" data center

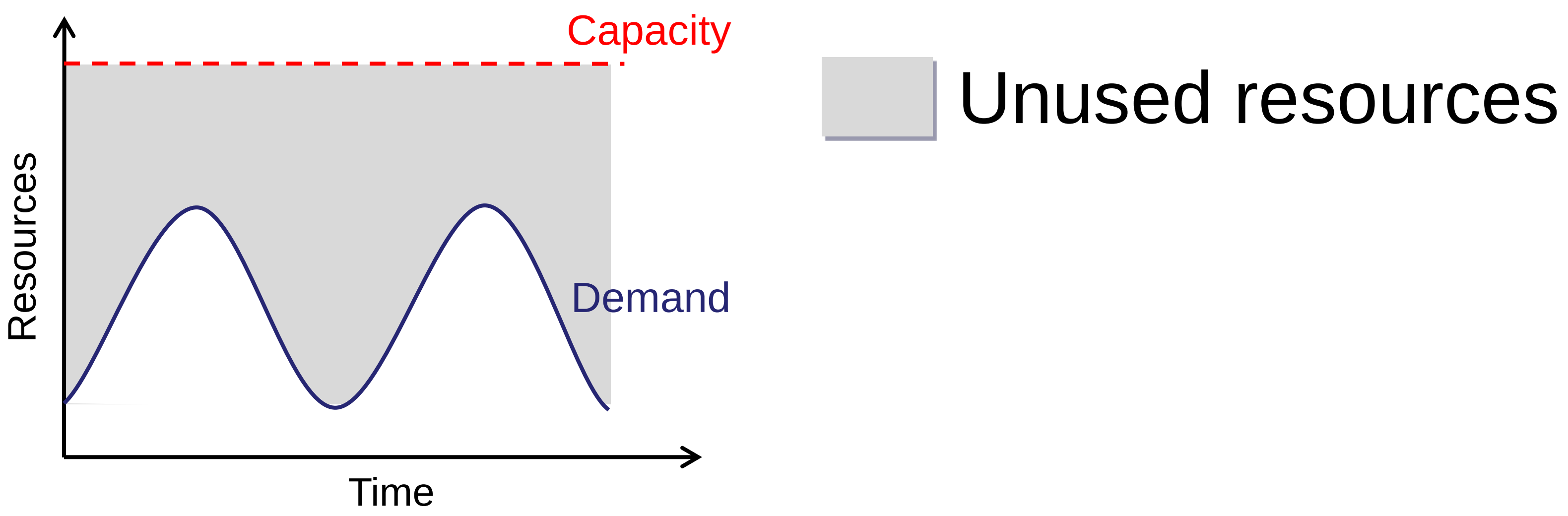


"Virtual" data center in the cloud

Unused resources

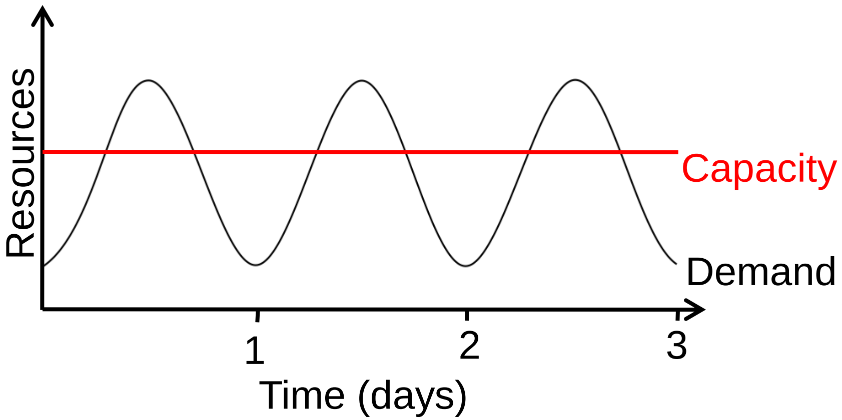
Risk of Under Utilization

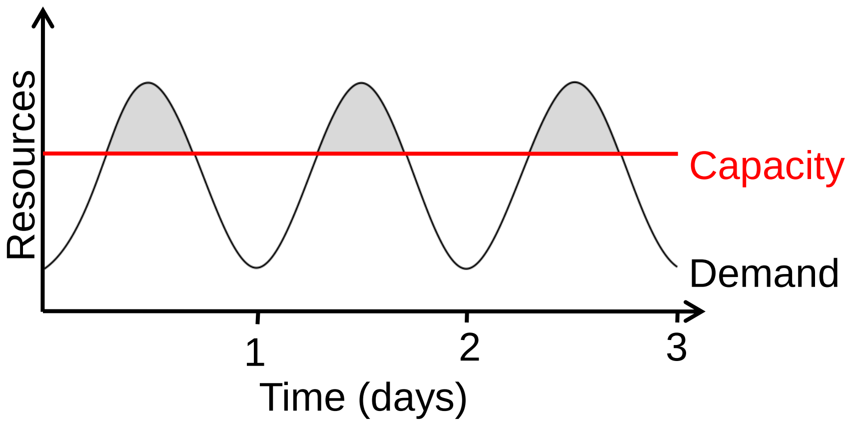
- Underutilization results if “peak” predictions are too optimistic



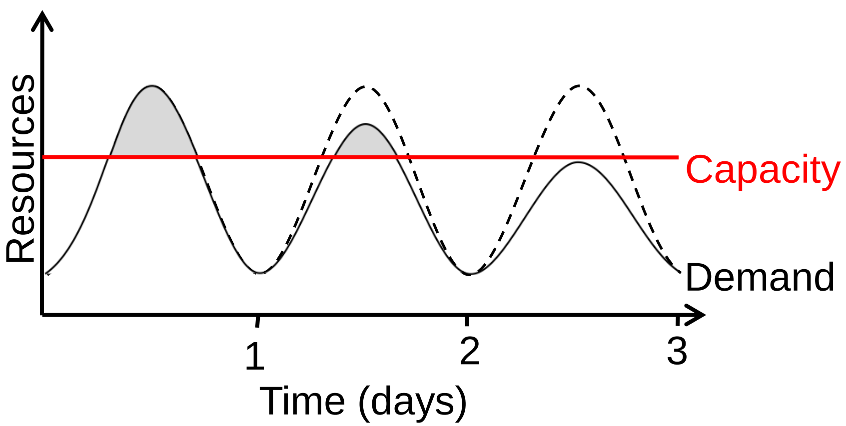
Static data center

Risks of Under Provisioning





Lost revenue



Lost users

New Scenarios Enabled by "Risk Transfer" to Cloud

- Not (just) Capital Expense vs. Operation Expense!
- "Cost associativity": 1,000 CPUs for 1 hour same price as 1 CPUs for 1,000 hours (@$0.08/hour)
  - RAD Lab graduate students demonstrate improved Hadoop (batch job) scheduler—on 1,000 servers
- *Major enabler* for SaaS startups
  - *Animoto* traffic doubled every 12 hours for 3 days when released as Facebook plug-in
  - Scaled from 50 to >3500 servers
  - ***...then scaled back down***
- Gets IT gatekeepers out of the way
  - not unlike the PC revolution

Hybrid / Surge Computing

- Keep a local "private cloud" running same protocols as public cloud
- When need more, "surge" onto public cloud, and scale back when need fulfilled
- Saves capital expenditures by not buying and deploying power distribution, cooling, machines that are mostly idle

What Scientists Don't Get about Cloud Computing

- Economic Analysis: Cost to buy a cluster assuming run 24x7 for 3 years vs. cost of same number of hours on Cloud Computing
- Ignores:
  - Cost of science grad student as sys. admin. (mistakes, negative impact on career, …)
  - Cost (to campus) of space, power, cooling
  - Opportunity cost of waiting when in race to be first to publish results: 20 local servers for a year vs. 1000 cloud servers for a week

Energy & Cloud Computing?

- Cloud Computing saves Energy?
- Don't buy machines for local use that are often idle
- Better to ship bits as photons over fiber vs. ship electrons over transmission lines to convert via local power supplies to spin disks and power processors and memories
  - Clouds use nearby (hydroelectric) power
  - Leverage economies of scale of cooling, power distribution

Energy & Cloud Computing?

- Techniques developed to stop using idle servers to save money in Cloud Computing can also be used to save power
  - Up to Cloud Computing Provider to decide what to do with idle resources
- New Requirement: Scale DOWN and up
  - Who decides when to scale down in a datacenter?
  - How can Datacenter storage systems improve energy?

Challenges & Opportunities

- "Top 10" Challenges to adoption, growth, & business/policy models for Cloud Computing
- Both technical and nontechnical
- Most translate to 1 or more *opportunities*
- Complete list in paper
- Paper also provides worked examples to quantify tradeoffs ("Should I move my service to the cloud?")

Growth Challenges

Challenge	Opportunity
Programming for large distributed systems	SEJITS – See Armando Fox talk at 1:30 in Room 1927
Scalable structured storage	Major research opportunity
Scaling quickly	Invent Auto-Scaler that relies on ML; Snapshots
Performance unpredictability	Improved VM support, flash memory, scheduling VMs
Data transfer bottlenecks	FedEx-ing disks, Data Backup/Archival

Adoption Challenges

Challenge	Opportunity
Availability / business continuity	Multiple providers & Multiple Data Centers
Data lock-in	Standardization
Data Confidentiality and Auditability	Encryption, VLANs, Firewalls; Geographical Data Storage

Policy and Business Challenges

Challenge	Opportunity
Reputation Fate Sharing	Offer reputation-guarding services like those for email
Software Licensing	Pay-as-you-go licenses; Bulk licenses

Outline

- What is Cloud Computing?
- Software as a Service / Cloud Computing in Education at UC Berkeley
- UC Berkeley RAD Lab Research Program in Cloud Computing
- Q&A

Software Education in 2010 (or: the case for teaching SaaS)

- Traditional "depth first" CS curricula vs. Web 2.0 breadth
  - Databases, Networks, OS, SW Eng/Languages, Security, ...
  - Students want to write Web apps,learn bad practices by osmosis
  - Medium of instruction for SW Eng. courses not tracking languages/tools/techniques actually in use
- New: languages & tools are actually good now
  - Ruby, Python, etc. are *tasteful* and allow reinforcing *important CS concepts* (higher-order programming, closures, etc.)
  - tools/frameworks enable *orders of magnitude* higher productivity than 1 generation ago, including for ***testing***
- Great fit for ugrad education
  - Apps can be developed & deployed on semester timescale
  - Relatively rapid gratification => projects outlive the course
  - Valuable skills: most industry SW moving to SaaS

Comparison to other SW Eng./programming courses

- Open-ended project
  - vs. “fill in blanks” programming
- Focus on SaaS
  - vs. Android, Java desktop apps, etc.
- Focus on RoR as high-level framework
- Projects expected to *work*
  - vs. working pieces but no artifact
  - most projects actually do work, some continue life outside class
- Focus on how “big ideas” in languages/programming enable high productivity

Web 2.0 SaaS as Course Driver

- Majority of students: ability to design own app was key to appeal of the course
  - design things they or their peers would use
- High productivity frameworks => projects *work*
  - actual gratification from using CS skills, vs. getting N complex pieces of Java code to work but not integrate
- Fast-paced semester is good fit for agile iteration-based design
- Tools used are same as in industry

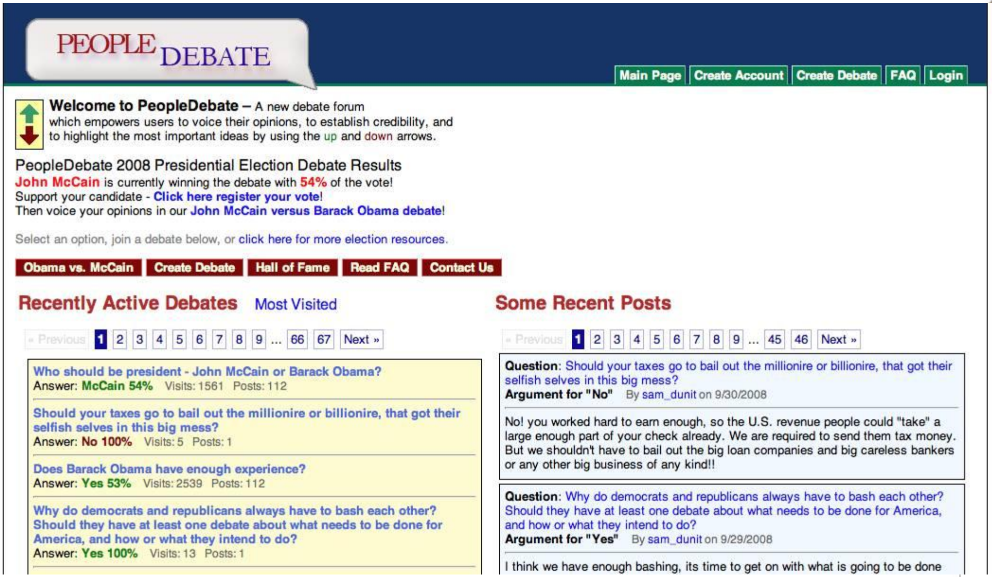
Cloud Computing as a Supporting Technology

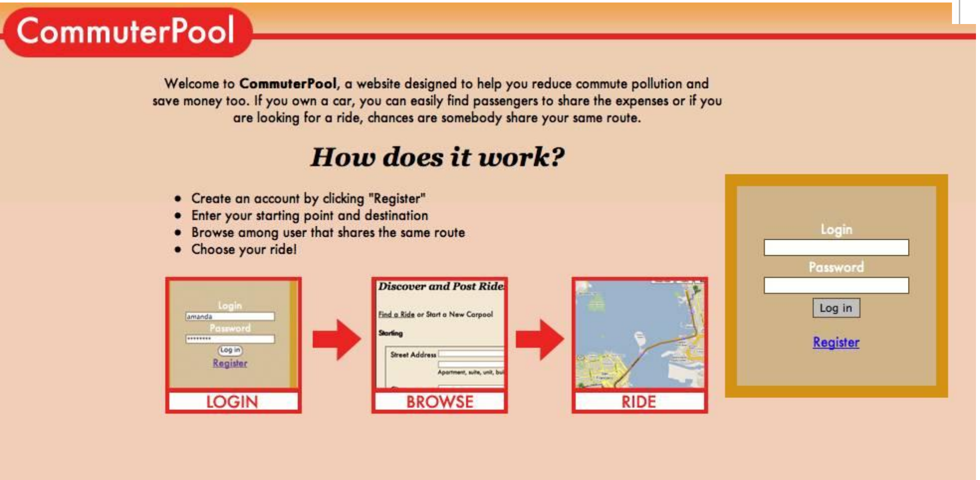
- Elasticity is great for courses!
  - Watch a database fall over: ~200 servers needed
  - Lab deadlines, final project demos don't collide
  - Donation from AWS; even more cost effective
- VM image simplifies courseware distribution
  - Prepare image ahead of time
  - Students can be *root* if need to install weird SW, libs...
- Students get better hardware
  - cloud provider updates HW more frequently
  - cost associativity
- VM images compatible with Eucalyptus—enables hybrid cloud computing

Moving to cloud computing

What	Before	After
Compute servers	4 nodes of R cluster	EC2
Storage	local Thumper	S3, EBS
Authentication	login per student, MySQL username/tables per student, ssh key for SVN per student	EC2 keypair + Google account
Database	Berkeley ITS shared MySQL	MySQL on EC2
Version control	local SVN repository	Google Code SVN
Horizontal scaling	???	EC2 + haproxy/nginx
Software stack management	burden Jon Kuroda	create AMI

SaaS Course Success Stories





Success stories, cont.

- Fall 2009 project: matching undergrads to research opportunities
- Fall 2009 project: Web 2.0 AJAXy course scheduler with links to professor reviews
- Spring 2010 projects: apps to stress RAD Lab infrastructure
  - gRADit: vocabulary review as a game
  - RADish: comment filtering taken to a whole new level

SaaS Student Feedback

- Comment from alum who took traditional Software Engineering Course (in Java) : “SaaS Project would have taken more than 2x the time in Java”
- Comment from instructor of traditional SWE course: “most projects didn’t really work at the end”
- **Hard to be as productive at lower level of abstraction than Ruby on Rails**

Moving to cloud computing

What	Before	After
Compute servers	4 nodes of R cluster	EC2
Storage	local Thumper	S3, EBS
Authentication	login per student, MySQL username/tables per student, ssh key for SVN per student	EC2 keypair + Google account
Database	Berkeley ITS shared MySQL	MySQL on EC2
Version control	local SVN repository	Google Code SVN
Horizontal scaling	No (Can't afford it)	EC2 + haproxy/nginx
Software stack management	burden local systems administrator	create AMI

SaaS Changes Demands on Instructional Computing?

- Runs on your laptop or class account
- Good enough for course project
- Project scrapped when course ends
- Intra-class teams
- Courseware: tarball or custom installs
- Code never leaves UCB

- Per-student/per-course account

- Runs in cloud, remote management
- Your friends can use it => *ilities matter
- Gain customers => app outlives course
- Teams cross class & UCB boundaries
- Courseware: VM image
- Code released open source, résumé builder

- General, collaboration-enabling tools & facilities

Summary: Education

- Web 2.0 SaaS is a great motivator for teaching software skills
  - students get to build artifacts they themselves use
  - some projects continue after course is over
  - opportunity to (re-)introduce "big ideas" in software development/architecture
- Cloud computing is great fit for CS courses
  - elasticity around project deadlines
  - easier administration of courseware
  - students can take work product with them after course (e.g. use of Eucalyptus in RAD Lab)

Outline

- What is Cloud Computing?
- Software as a Service / Cloud Computing in Education at UC Berkeley
- UC Berkeley RAD Lab Research Program in Cloud Computing
- Q&A

RAD Lab 5-year Mission

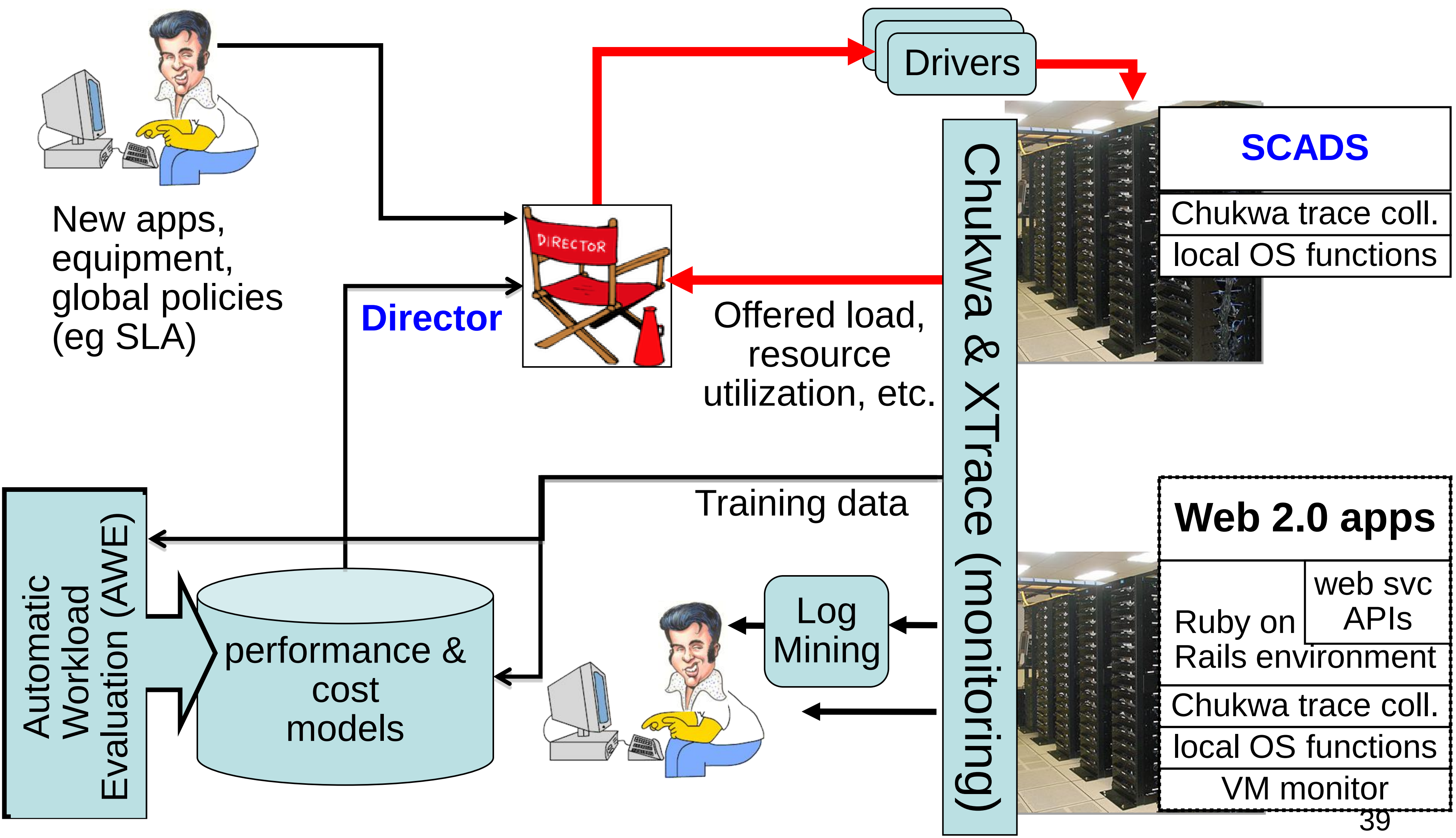
*Enable **1 person** to develop, deploy, operate next -generation Internet application*

- Key enabling technology: Statistical machine learning
  - debugging, power management, performance prediction, ...
- Highly interdisciplinary faculty & students
  - PI’s: Fox/Katz/Patterson (systems/networks), Jordan (machine learning), Stoica (networks & P2P), Joseph (systems/security), Franklin (databases)
  - 2 postdocs, ~30 PhD students, ~10 undergrads

Machine Learning & Systems

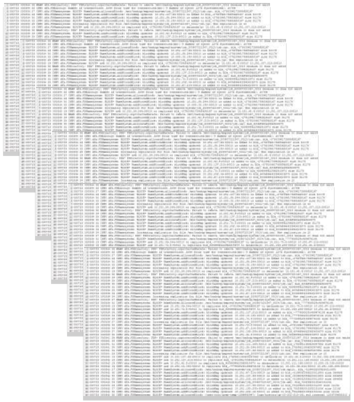
- **Recurring theme**: cutting-edge Statistical Machine Learning (SML) works where simpler methods have failed
  - Predict performance of complex software system when demand is scaled up
  - Automatically add/drop servers to fit demand, without violating Service Level Objective (SLO)
  - Distill millions of lines of log messages into an operator-friendly “decision tree” that pinpoints “unusual” incidents/conditions

RAD Lab Prototype: System Architecture



Console logs are not operator friendly

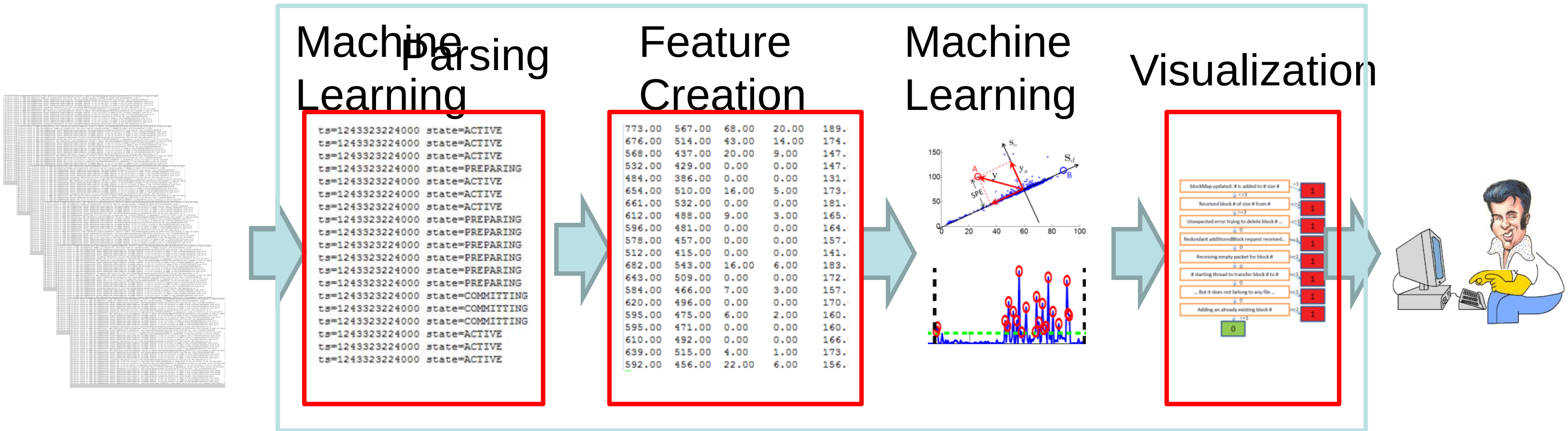
Console Logs



Operators

- Problem – Don't know what to look for!
  - Console logs are intended for a single developer
  - Assumption: log writer == log reader
  - Today many developers => massive textual logs
- Our goal - Discover the most interesting log messages without any prior input

Console logs are hard for machines too

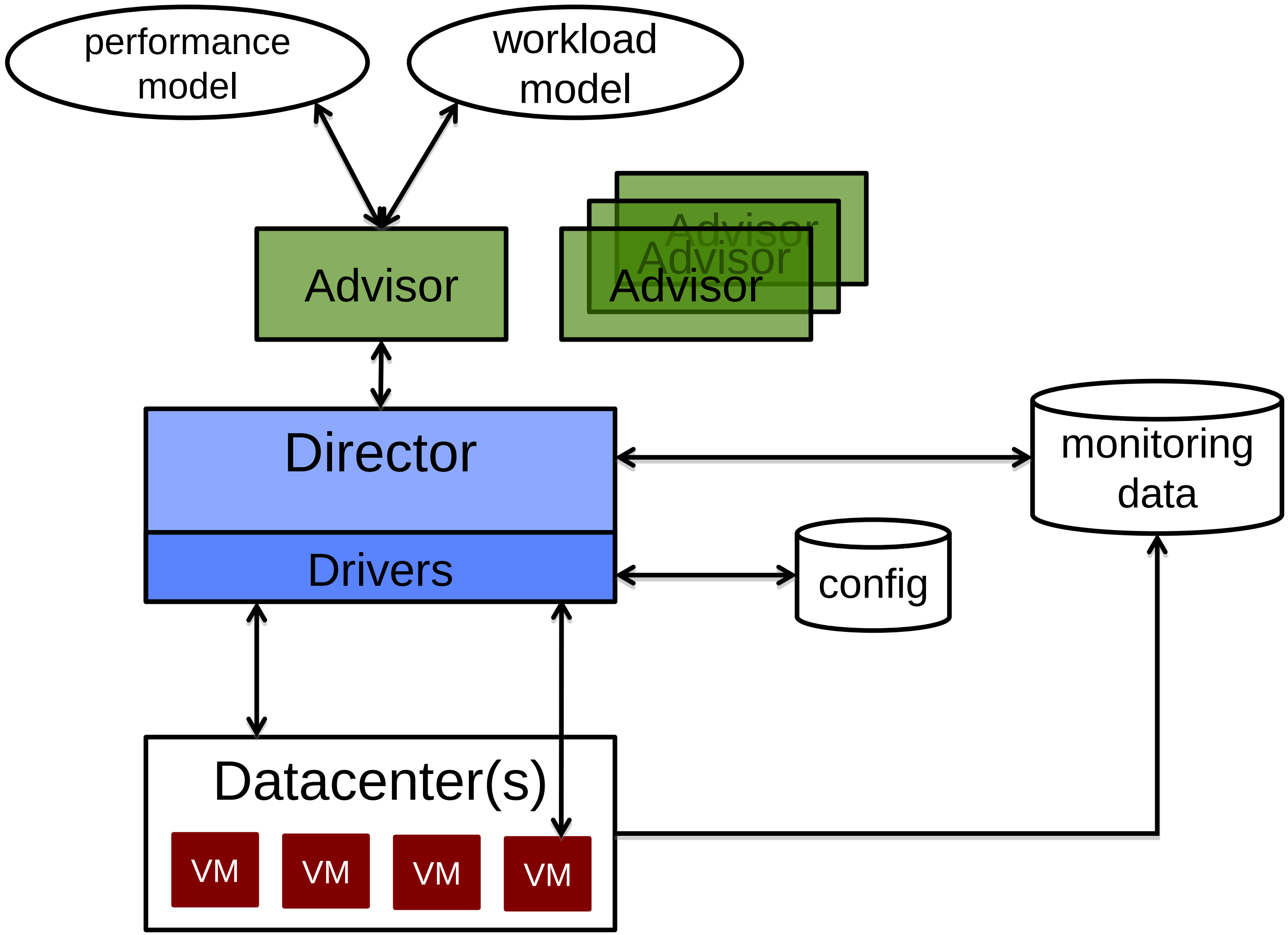


- Problem
  - Highly unstructured, looks like free text
  - Not able to capture detailed program state with texts
  - Hard for operators to understand detection results
- Our contribution
  - A general framework for processing console logs
  - Efficient parsing and features
  - 24M lines of log to 1 page picture of anamolies

Automatic Management of a Datacenter

- As datacenters grow, need to automatically manage the applications and resources
  - examples:
    - deploy applications
    - change configuration, add/remove virtual machines
    - recover from failures
- Director:
  - mechanism for executing datacenter actions
- Advisors:
  - intelligence behind datacenter management

Director Framework



Director Framework

- Director
  - issues low-level/physical actions to the DC/VMs
    - request a VM, start/stop a service
  - manage configuration of the datacenter
    - list of applications, VMs, …
- Advisors
  - update performance, utilization metrics
  - use workload, performance models
  - issue logical actions to the Director
    - start an app, add 2 app servers

What About Storage?

- Easy to imagine how to scale up and scale down computation
- Database don’t scale down, usually run into limits when scaling up
- What would it mean to have datacenter storage that could scale up and down as well so as to save money for storage in idle times?

SCADS: Scalable, Consistency-Adjustable Data Storage

- Goal: Provide web application developers with *scale independence* as site grows
  - No changes to application
  - Cost / User doesn't increase as users increase
  - Latency / Request doesn't increase as users↑
- Key Innovations
  - Performance safe query language (PIQL)
  - Declarative performance/consistency tradeoffs
  - Automatic scale up and down using machine learning (Director/Advisor)

Conclusion

- Cloud Computing will transform IT industry
  - Pay-as-you-go utility computing leveraging economies of scale of Cloud provider
  - Anyone can create/scale next eBay, Twitter…
- Transform academic research, education too
- Cloud Computing offers $ for systems to scale down as well as up: save energy too
- RAD Lab addressing New Cloud Computing challenges: SEJITS, Director to manage datacenter using SML, Scalable DC Store

Backup Slides

UCB SaaS Courses

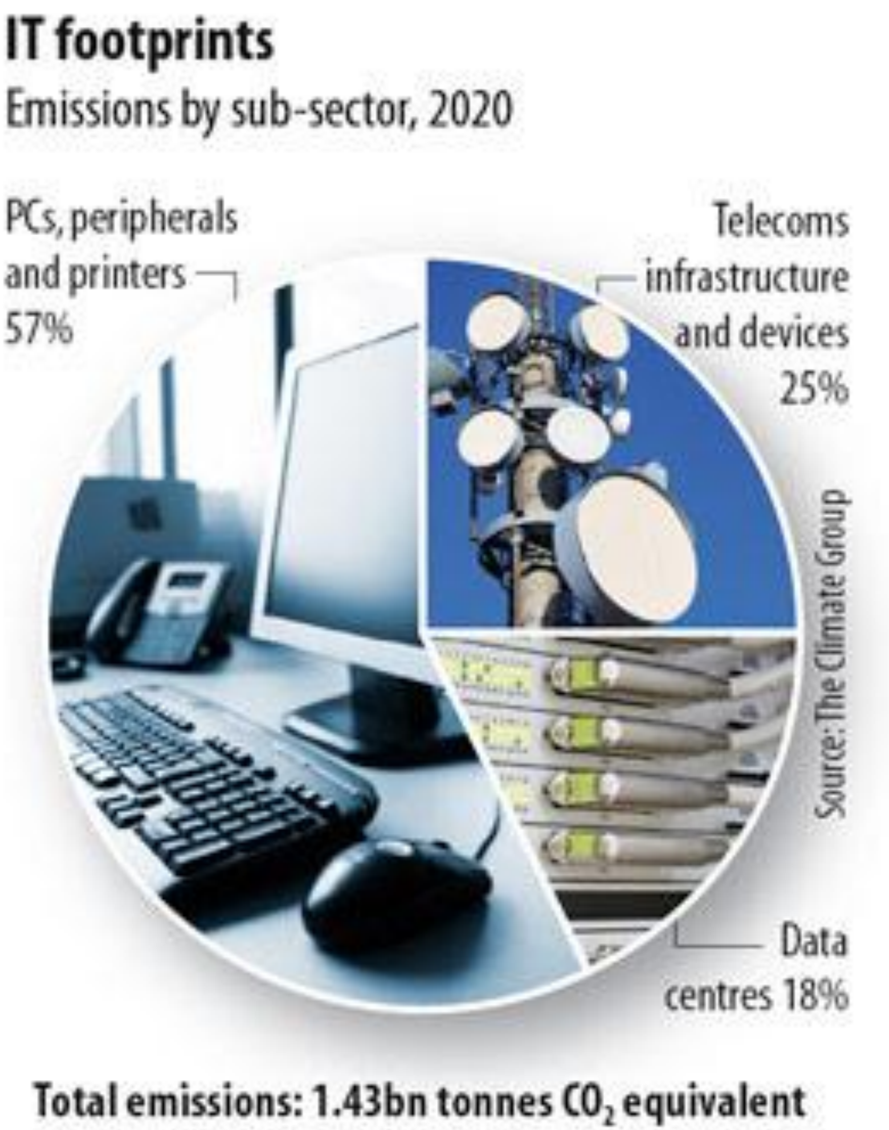
	Lower div.	Upper div.	Grad.
Understand Web 2.0 app structure	✔		
Understand high-level abstraction toolkits like RoR	✔	✔	
How high-level abstractions implemented		✔	✔
Scaling/operational challenges of SaaS		✔	✔
Develop & deploy SaaS app	✔	✔	
Implement new abstractions, languages, or analysis techniques for SaaS			✔

2020 IT Carbon Footprint

820m tons CO_2

2007 Worldwide IT carbon footprint:
2% = 830 m tons CO_2
Comparable to the global aviation industry

Expected to grow to 4% by 2020



360m tons CO_2

260m tons CO_2