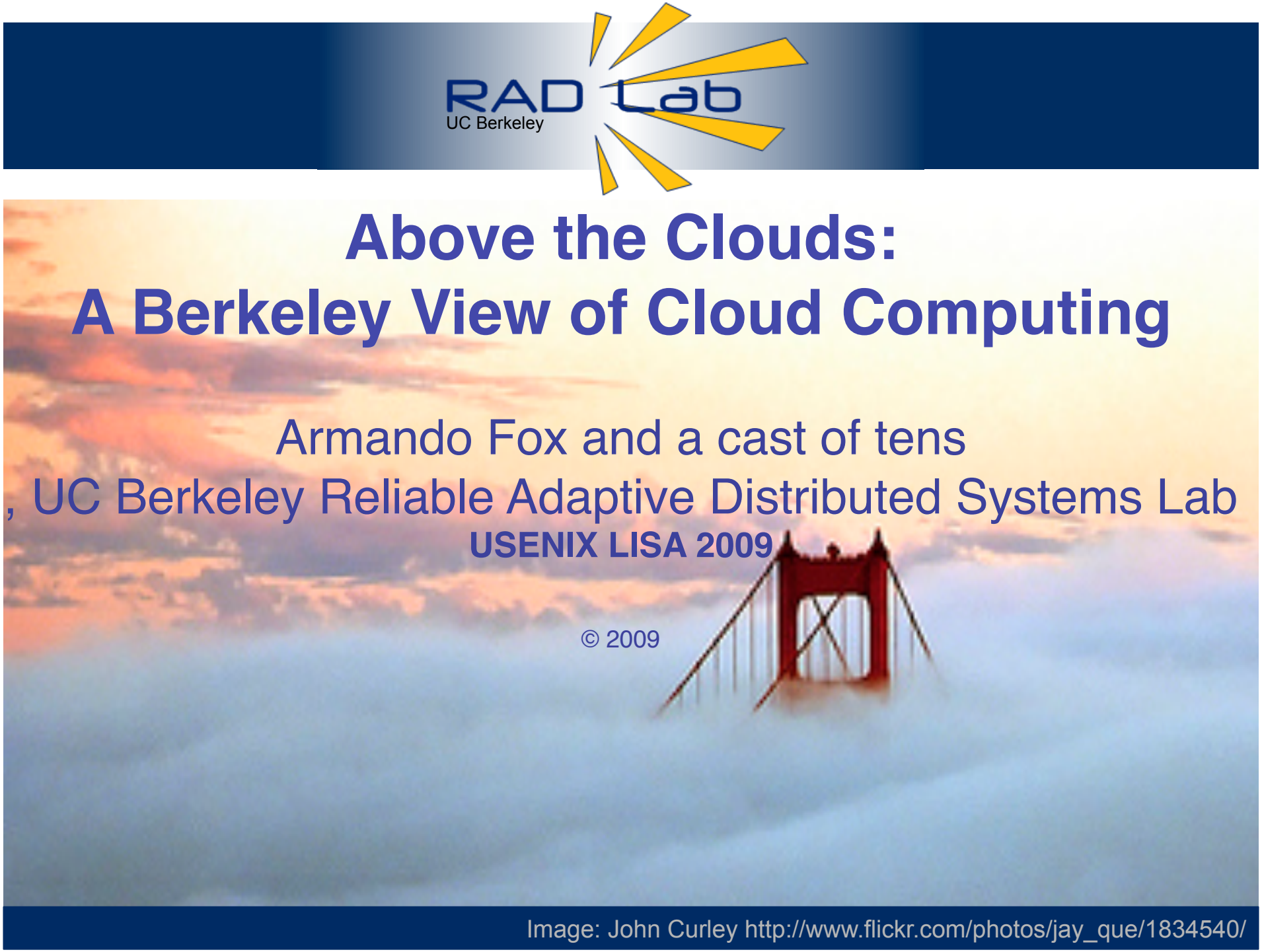


Above the Clouds: A Berkeley View of Cloud Computing

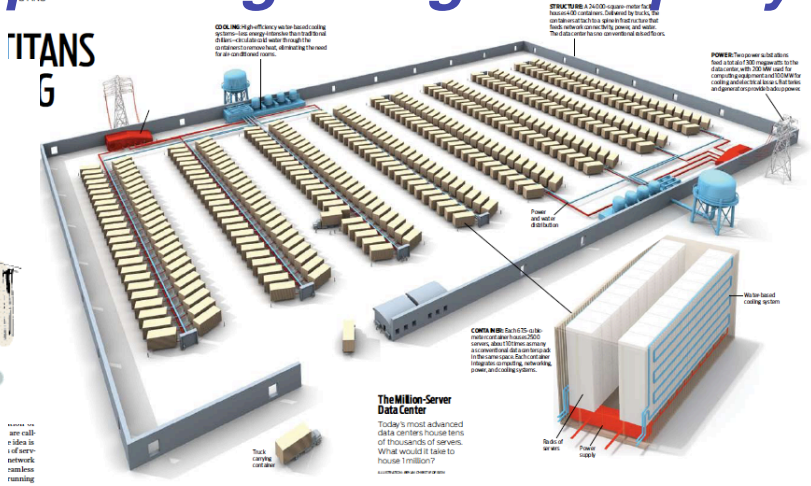
Armando Fox and a cast of tens
, UC Berkeley Reliable Adaptive Distributed Systems Lab

USENIX LISA 2009

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

Datacenter is new "server"

- *"Program"* == Web search, email, map/GIS, …
- *"Computer"* == 1000's computers, storage, network
- Warehouse-sized facilities and workloads
- New datacenter ideas (2007-2008): truck container (Sun), floating (Google), In Tents Computing (Microsoft)
- ***How to enable innovation in new services without first building & capitalizing a large company?***



photos: Sun Microsystems & datacenterknowledge.com

RAD Lab 5-year Mission

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

- Key enabling technology: Statistical machine learning
  - management, scaling, anomaly detection, performance prediction...
- interdisciplinary: 7 faculty, ~30 PhD's, ~6 ugrads, ~1 sysadm
- Regular engagement with industrial affiliates keeps us from smoking our own dope too often

How we got into the clouds

- **Theme**: cutting-edge statistical machine learning works where simple methods fail
  - Resource utilization prediction
  - Adding/removing storage bricks to meet SLA
  - Console log analysis for problem finding

- **Sponsor feedback**: Great, now show that it works on *at least* 1000's of machines

Utility Computing to the Rescue: Pay as you Go

- Amazon Elastic Compute Cloud (EC2)
- "Compute units" ~~$0.10-0.80/hr.~~ $0.085/hr & up
  - 1 CU ≈ 1.0-1.2 GHz 2007 AMD Opteron/Xeon core

"Instances"	Platform	Cores	Memory	Disk
Small - $0.085 / hr	32-bit	1	1.7 GB	160 GB
Large - $0.34/ hr	64-bit	4	7.5 GB	850 GB – 2 spindles
XLarge - $0.68/ hr	64-bit	8	15.0 GB	1690 GB – 3 spindles
Options....extra memory, extra CPU, extra disk, ...				

- storage (~0.15/GB/month)
- network (~0.10-0.15/GB external; 0.00 internal)
- Everything virtualized, even concept of independent failure

Cloud Computing is Hot *sigh*

"...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." *Sept. 2008*

"We've been building data center after data center, acquiring application after application, ...driving up the cost of technology immensely across the board. We need to find a more innovative path." *Sept. 2009*

A Berkeley View of Cloud Computing

abovetheclouds.cs.berkeley.edu

- 2/09 White paper by RAD Lab PI's/students
- Goal: stimulate discussion on *what's new*
  - Clarify terminology
  - Quantify comparisons
  - Identify challenges & opportunities
- UC Berkeley perspective
  - industry engagement but no axe to grind
  - users of CC since late 2007

Rest of talk

1. What is it? What's new?
2. Challenges & Opportunities
3. "We should cloudify our datacenter/cluster/whatever!"
4. Academics in the cloud

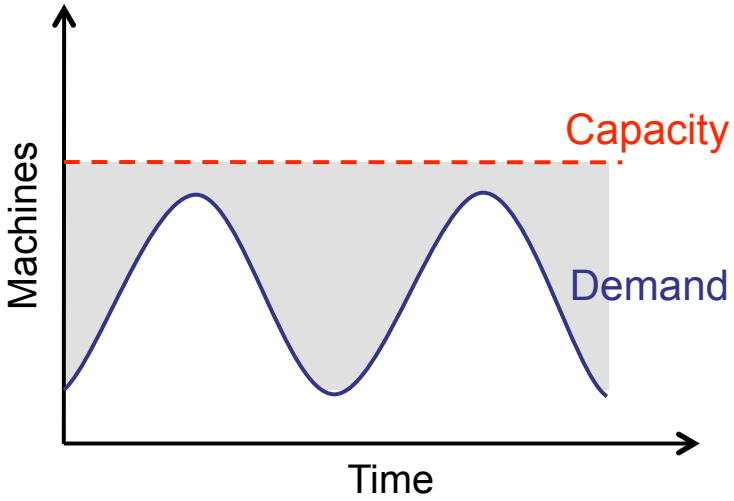
1. What is it? What's new?

- Old idea: Software as a Service (SaaS), predates Multics

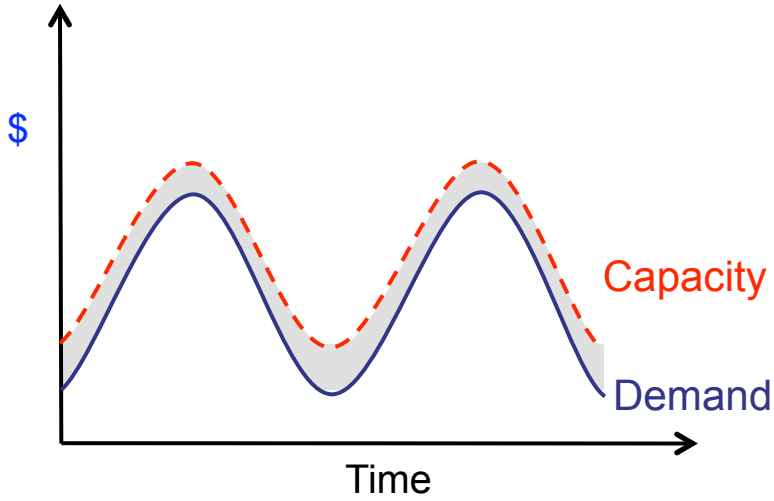
- **New:** pay-as-you-go, *utility computing*
  - Illusion of infinite resources on demand (minutes)
  - Fine-grained billing: release == don't pay
  - No minimum commitment
  - Earlier examples (Sun, Intel): longer commitment, more $$$/hour, no storage

Cloud Economics 101

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



"Statically provisioned" data center

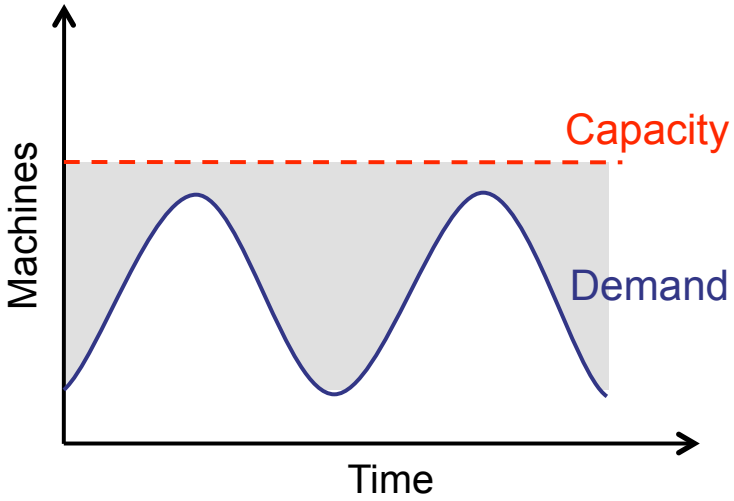


"Virtual" data center in the cloud

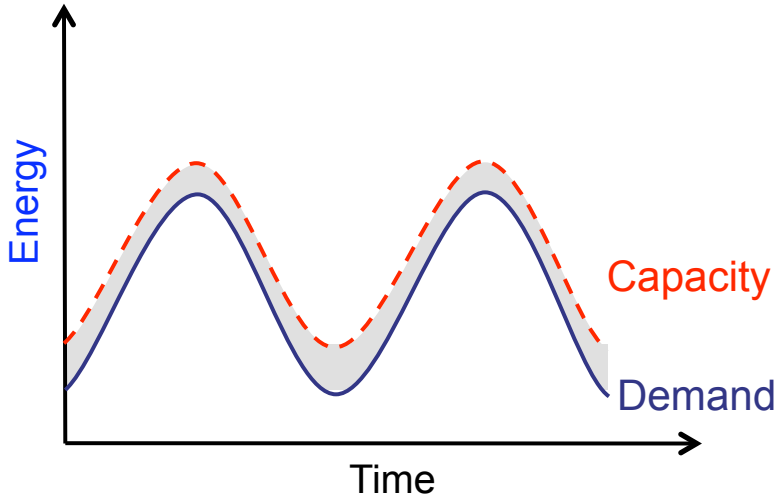
Unused resources

Cloud Economics 101

- Cloud Computing Provider: Could save energy



"Statically provisioned" data center



Real data center in the cloud

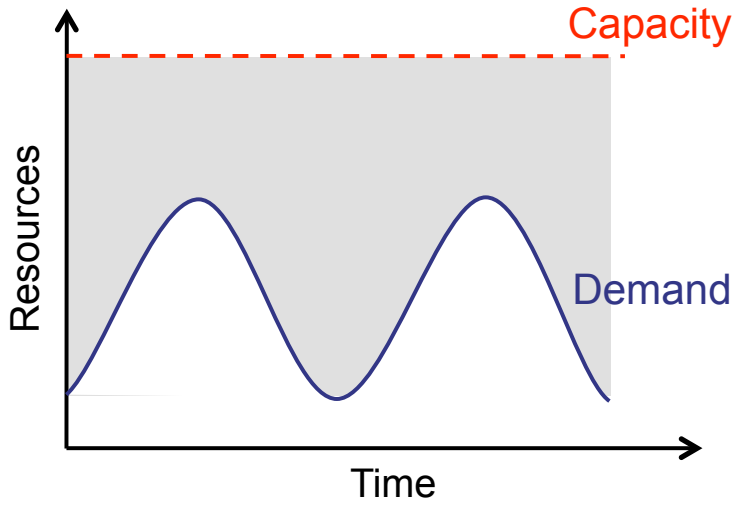
Unused resources

Back of the envelope

- Server utilization in datacenters: 5-20%
  - peaks 2x-10x average
- C = cost/hr. to use cloud (.085 for AWS)
- B = cost/hr. to buy server
  - $2K server, 3-year depreciation: $0.076
- HW savings = (peak/average util.) – (C/B)
  - in this example, save $$ if peak > 1.1x average
  - can also factor in network & storage costs
- Caveat: IT accounting often not so simple

Risk of Overprovisioning

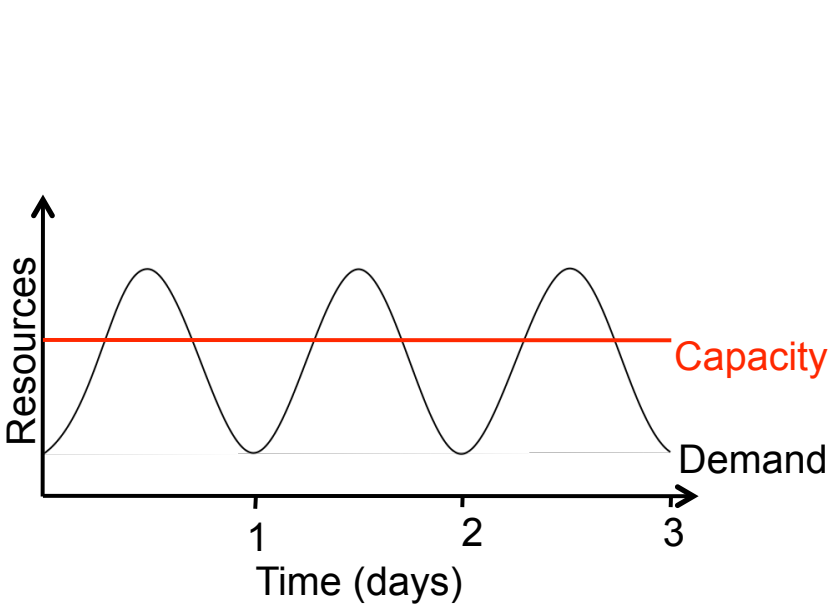
- Underutilization results if “peak” predictions are too optimistic

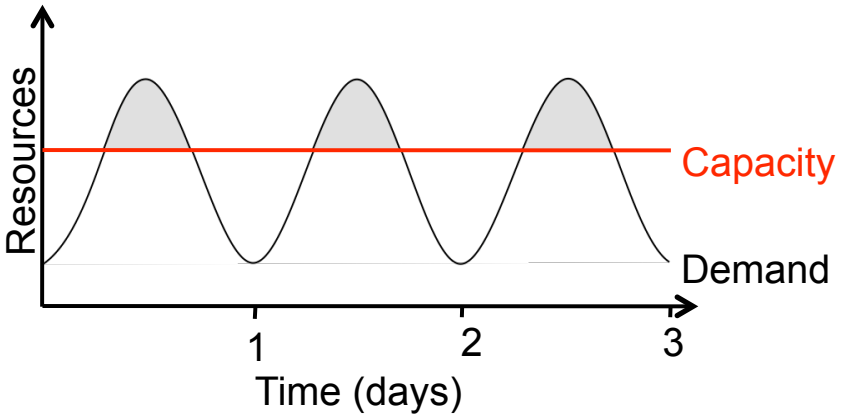


Unused resources

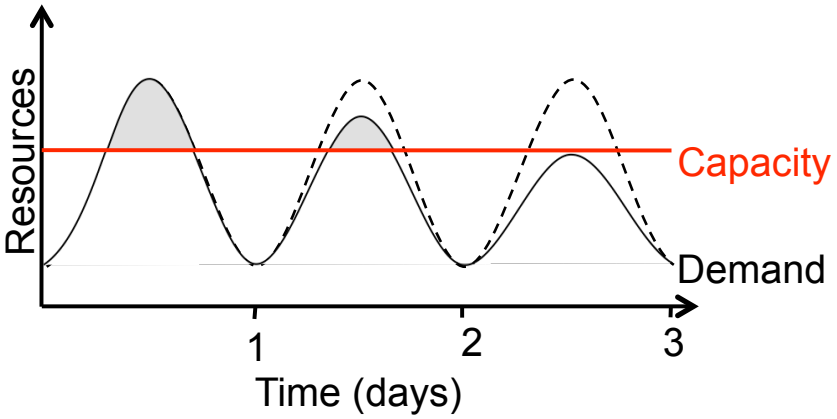
Static data center

Risks of Under Provisioning





Lost revenue



Lost users

Risk Transfer vs. CapEx/OpEx

- Over long timescales, a dollar is a dollar
- CC is *not* necessarily cheaper, esp. if you have steady, known capacity needs
- But *risk transfer* opens fundamentally new opportunities.

Risk Transfer: new scenarios

- "Cost associativity":
  1K servers x 1 hour == 1 server x 1K hours
  - Washington Post: Hillary Clinton's travel docs posted to WWW **<1 day** after released
  - RAD Lab: publish results on 1,000+ servers
- Major enabler for SaaS startups
  - *Animoto* Facebook plugin => traffic doubled every 12 hours for 3 days
  - Scaled from 50 to >3500 servers
  - ***...then scaled back down***

Why Now (not then)?

- Build-out of extremely large datacenters (10,000s ***commodity*** PCs)
- ...and how to run them
  - Infrastructure SW: e.g., Google File System
  - Operational expertise: failover, DDoS, firewalls...
  - economy of scale: 5-7x cheaper than provisioning medium-sized (100s/low 1000s machines) facility
- Necessary-but-not-sufficient factors
  - pervasive broadband Internet
  - Commoditization of HW & Fast Virtualization
  - Standardized (& free) software stacks

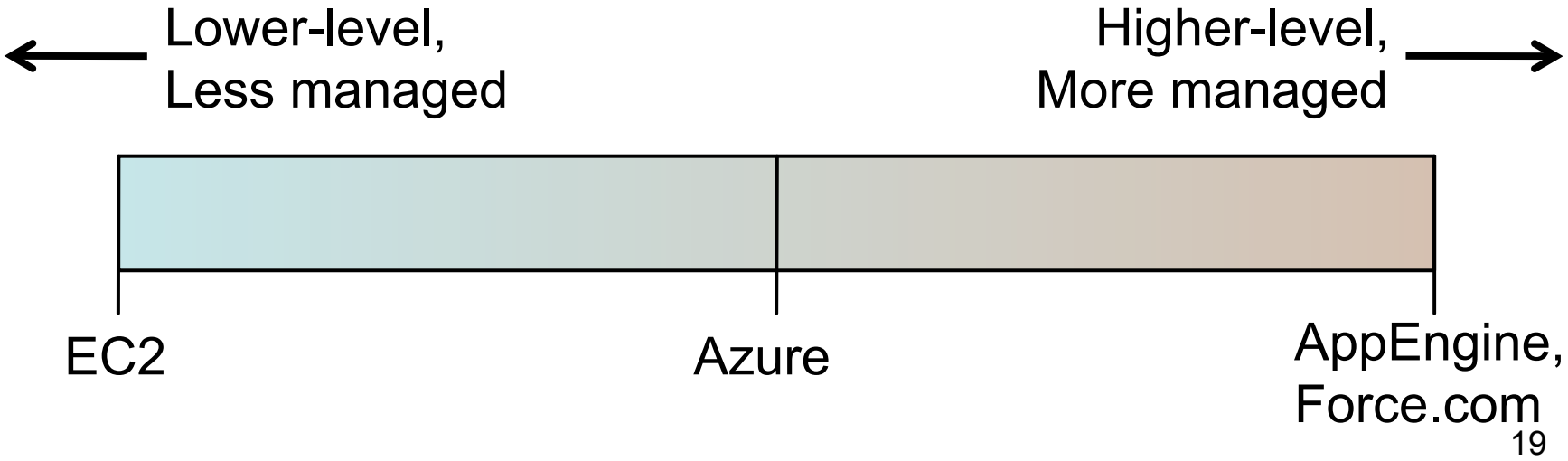
2. Challenges & Opportunities

A subset of what's in the paper

Both technical & nontechnical

Classifying Clouds

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



Lock-in/business continuity

Challenge	Opportunity
Availability / business continuity	Multiple providers & datacenters Open API's

- Few enterprise datacenters' availability is as good
- "Higher level" (AppEngine, Force.com) vs. "lower level" (EC2) clouds include proprietary software
  - \+ richer functionality, better built-in ops support
  - – structural restrictions
- FOSS reimplementations on way? (eg AppScale)

Data lock-in

Challenge	Opportunity
Data lock-in	Standardization

- FOSS implementations of storage (eg HyperTable)

- 10/19/09: Google Data Liberation Front

Data is a Gravity Well

Challenge	Opportunity
Data transfer bottlenecks	FedEx-ing disks, Data Backup/Archiving

- Amazon now provides “FedEx a disk” service
- and hosts free public datasets to “attract” cycles

Data is a Gravity Well

Challenge	Opportunity
Scale-up/scale-down structured storage	Major research opportunity

•Profileration of *non-relational* scalable storage:
SQL Services (MS Azure), Hypertable, Cassandra, HBase, Amazon SimpleDB & S3, Voldemort, CouchDB, NoSQL movement

Policy/Business Challenges

Challenge	Opportunity
Reputation Fate Sharing	Offer reputation-guarding services like those for email

4/2/09: FBI raid on Dallas datacenter shuts down legitimate businesses along with criminal suspects

10/28/09: Amazon will whitelist elastic-IP addresses and selectively raise limit on outgoing SMTP

Policy/Business Challenges

Challenge	Opportunity
Software Licensing	Pay-as-you-go licenses; Bulk licenses

2/11/09: IBM pay-as-you-go Websphere, DB2, etc. on EC2

Windows on EC2

FOSS makes this less of a problem for some potential cloud users

3. Should I cloudify?

Public vs. private clouds won't see same benefits

Benefit	Public	Private
Economy of scale	Yes	No
Illusion of infinite resources on-demand	Yes	Unlikely
Eliminate up-front commitment by users*	Yes	No
True fine-grained pay-as-you-go **	Yes	??
Better utilization (workload multiplexing)	Yes	Depends on size**
Better utilization & simplified operations through virtualization	Yes	Yes

* What about nonrecoverable engineering/capital costs?

** Implies ability to meter & incentive to release idle resources

Consider getting best of both with ***surge computing***

So, should I cloudify?

- Why? Is cost savings expected?
  - economies of scale unlikely for most shops
  - beware “double paying” for bundled costs
- Internal incentive to release unused resources?
  - If not...don’t expect improved utilization
  - Implies ability to meter (technical) and charge (nontechnical)

IT best practices become critical

- Authentication, data privacy/sensitivity
  - Data flows over public networks, stored in public infrastructure
  - Weakest link in security chain == ?
- Support/lifecycle costs vs. alternatives
  - Strong appliance market (e.g. spam filters)
  - “Accountability gap” for support

Hybrid/Surge Computing

- Use cloud for separate/one-off jobs?
- Harder: Provision steady state, overflow your app to cloud?
  - implies high degree of location independence, software modularity
  - must overcome most Cloud obstacles
  - FOSS reimplementations (Eucalyptus) or commercial products (VMware vCloud)?

Do my apps make sense in cloud?

- Some app types compelling
  - Extend desktop apps into cloud: Matlab, Mathematica; soon productivity apps?
  - Web-like apps with reasonable database strategy
  - Batch processing to exploit cost associativity, e.g. for business analytics
- Others cloud-challenged
  - Bulk data movement expensive, slow
  - Jitter-sensitive apps (long-haul latency & virtualization-induced performance distortion)

4. Academics in the Cloud: some experiences

(thanks: Jon Kuroda, Eric Fraser, Mike Howard)

Clouds in the RAD Lab

- Eucalyptus on ~40-node cluster
- Lots of Amazon AWS usage
- Workload can overflow from one to the other (same tools, VM images, ...)
- *Primarily for research/experiments that don't need to tie in with, eg, UCB Kerberos*
- *Permissions, authentication, access to home dirs from AWS, etc.—open problems*

An EECS-centric view

- Higher quality research
  - routinely do experiments on 100+ servers
  - many results published on 1,000+ servers
  - unthinkable a few years ago
- Get results faster => solve new problems
  - lots of machine learning/data mining research
  - eg console log analysis [Xu et al, SOSP 09 & ICDM 09]: minutes vs. hours means can do in near-real-time
- Save money? um...that was a non-goal

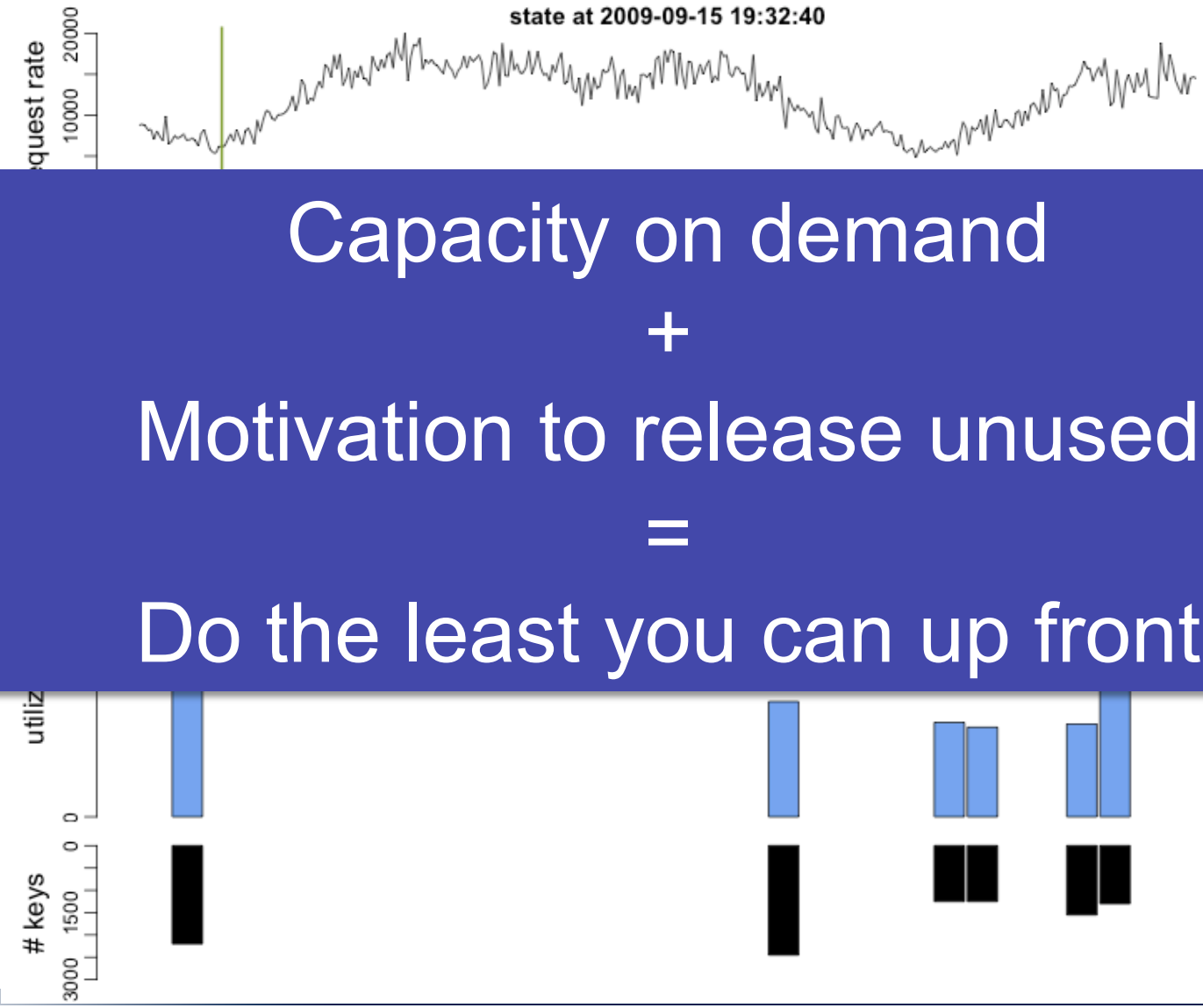
Obstacles to CC in Research

- Accounting models that reward cost-effective cloud use
- Funding/grants culture hasn't caught up to "CapEx vs. OpEx"
- Tools still require high sophistication
  - but attractive role for software appliances
- Software licensing isn't "cost associative"
  - typically still tied to seats or fixed #CPUs
  - less problematic for us as researchers

Cloud Computing & Statistical Machine Learning

- Before CC, performance optimization was mostly focused on small-scale systems
- CC ➔ detailed cost-performance model
  - Optimization more difficult with more metrics
- CC ➔ Everyone can use 1000+ servers
  - Optimization more difficult at large scale
- Economics rewards scale up **and down**
  - Optimization more difficult if add/drop servers
- SML↑ as optimization difficulty increases

Example: "elastic" key-value store for SCADS [Armbrust et al, CIDR 09]



Capacity on demand

+

Motivation to release unused

=

Do the least you can up front

CS education in the Cloud

- Moved Berkeley SaaS course to AWS
  - expose students to realistic environment
  - Watch a database fall over: would have needed 200 servers for ~20 project teams
  - End of term project demos, Lab deadlines
- VM image simplifies courseware distribution
  - Students can be **root**
  - repair damage == reinstantiate image

Summary: Clouds in EECS

- Focus is new research/teaching opportunities vs. cost savings
- Mileage may vary in other departments
- Tools still require sophistication
- Authentication, other “admino-technical” issues largely unsolved
- Funding/costing models not caught up

Wrapping up...

Summary: What's new

- CC "Risk transfer" enables new scenarios
  - Startups and prototyping
  - One-off tasks that exploit "cost associativity"
  - Research & education at scale
- Improved utilization and lower costs if scale down as well as up
  - Economic motivation to scale down
  - Changes thinking about load balancing, SW design to support scale-down

Summary: Obstacles

- How “dependent” can you become?
  - Data expensive to move, no universal format
  - Management API’s not yet standardized
  - Doesn’t (necessarily) eliminate reliance on proprietary SW
- SW licensing mostly cloud-unfriendly
- Security considerations, IT best practices
- Difficulty of quantifying savings
- Locus of administration/accountability?

Should I cloudify?

- Expecting to save money?
  - Economy of scale unlikely; savings more likely from better utilization
  - But must design for resource accounting & offer incentive to release
  - Does hybrid/surge make sense?
- Even if don't move to cloud…use as driver
  - enforce best practices
  - identify bundled costs => true cost of IT

Conclusion

Is cloud computing all hype?

No.

Is it a fad that will fizzle out?

We think it's a major sea change.

Is it for everyone?

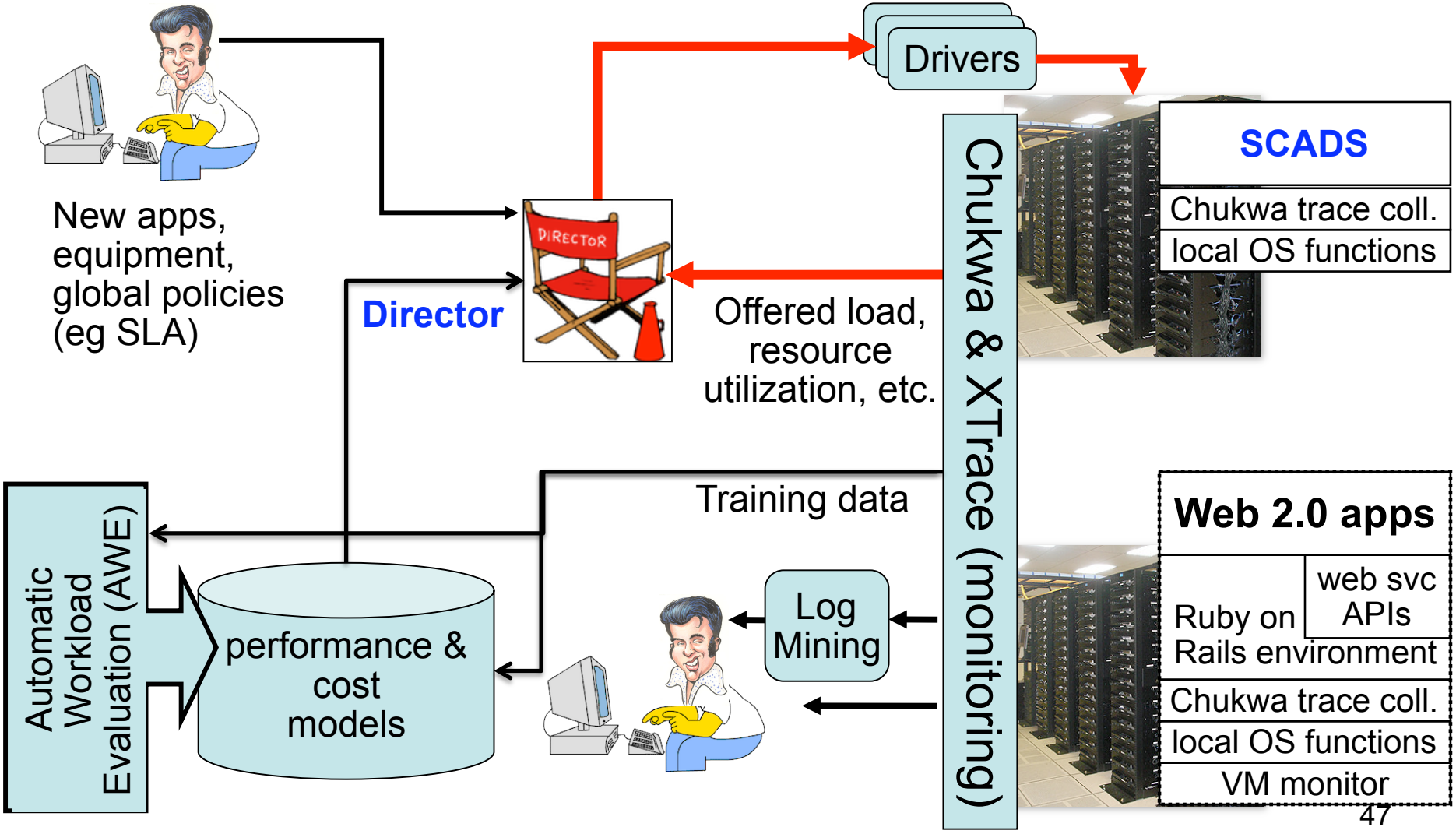
No/not yet, but be familiar with obstacles & opportunities

Thank you!

More: ***abovetheclouds.cs.berkeley.edu***

BACKUP SLIDES

RAD Lab Prototype: System Architecture



CC Changes Demands on Instructional Computing?

- Runs on your laptop or class Un*x account
- Good enough for course project
- project scrapped when course ends
- Intra-class teams
- Courseware: custom install
- Code never leaves UCB

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

- Per-student/per-course account

- General, collaboration-enabling tools & facilities

Big science in the cloud?

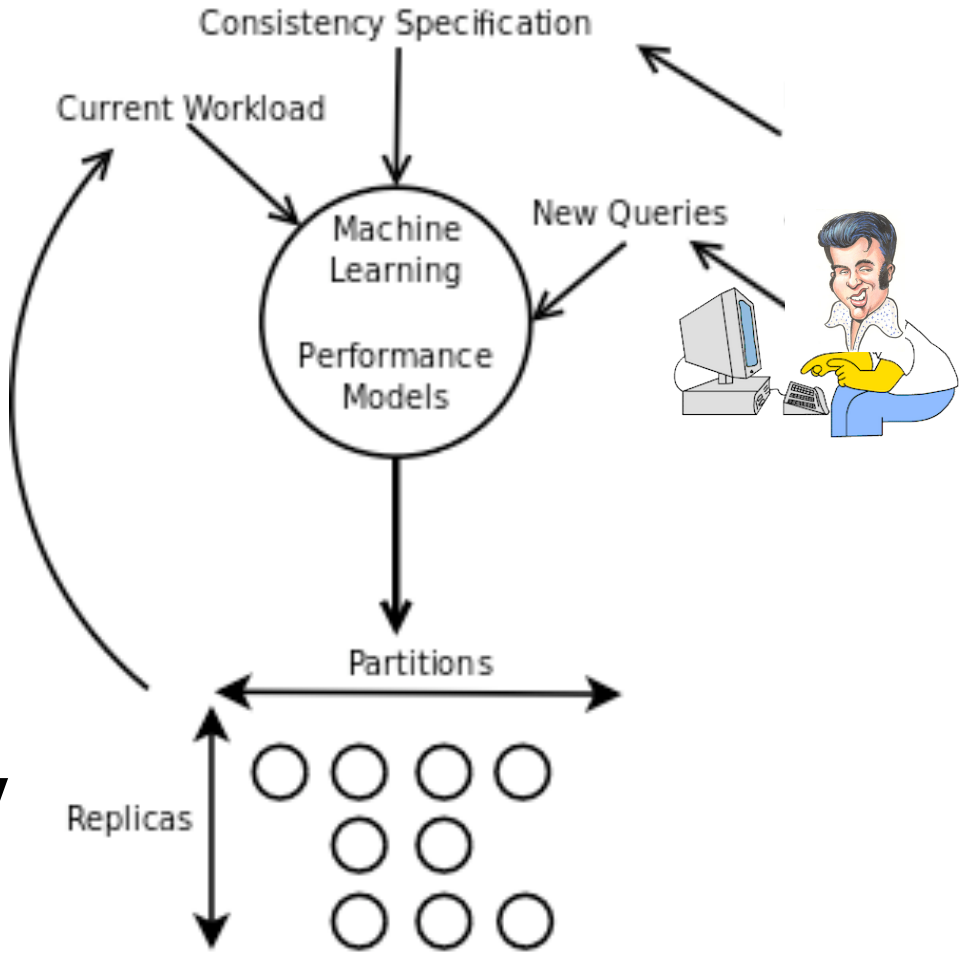
- Web apps restructured to “shared-nothing friendly” thru 90s; can science do same?
  - gang scheduling for clouds/virtual clouds?
  - rethink storage vs. checkpointing vs. code structure
  - move to much higher level languages (leave tuning to macroblocks/runtime, not woven into source code)
  - Data-intensive (I/O rates & volume) needs of science apps
- Opportunity for “cost associativity”!

SCADS: Scalable, Consistency-Adjustable Data Storage

- **Scale Independence –** as #users grows:
  - No changes to application
  - Cost per user doesn't increase
  - Request latency doesn't change
- **Key Innovations**
  1. Performance safe query language
  2. Declarative performance/consistency tradeoffs
  3. **Automatic scale up and down using machine learning**

Scale Independence Arch

- Developers provide performance safe queries along with consistency requirements
- Use ML, workload information, and requirements to provision proactively via repartitioning keys and replicas



SCADS Performance Model (on m1.small, all data in memory)