15-719/18-847b Advanced Cloud Computing

Garth Gibson
Majd Sakr
Greg Ganger

Mrigesh Kalvani
Jinliang Wei
Mengjin Yan
Qing Zheng

www.cs.cmu.edu/~15719
Post privately on Piazza to ask questions

Advanced Cloud Computing

- PhD level course covering range of cloud computing technologies
  - Majority of class will be masters students
  - All non-PhD students must have a B in 15-513/15-213/18-213/15-619
  - PhD students have different projects, and
    lead a Fri discussion session that is optional for non-PhD students
- Significant load of readings, primarily research papers
  - Read non-optional papers before class and ask questions
  - We will ask spot-graded questions on readings in class of specific students
- Emphasis on ability to make design and implementation choices,
  not primarily about using cloud computing

Syllabus

- What is cloud computing?
- Encapsulating computation
- Parallel programming frameworks
- Scheduling cloud clusters
- Elasticity
- Cloud storage
- Cloud edge platforms
- Fault tolerance vs availability
- Diagnosing & debugging cloud services
- Cloud networking
- Geo-replication
- Security
- Tail latency considerations

Logistics

- SCS students should be in 15-719, ECE students should be in 18-847b
  - Others (e.g. INI students) should be in whichever section has space (we'll send email)
- Communication – www.cs.cmu.edu/~15719
  - Check for changes often!
  - https://piazza.com/cmu/spring2017/15719/home is the primary Q&A mechanism
    - Post privately to instructors for questions; we'll post answers to all if appropriate
    - No access to piazza without prereqs or instructor approval
- Readings – typically 1-2 papers per class meeting; read before class!
  - See web syllabus for readings (access from a CMU IP address)
- Meetings – Mon & Wed A302 DH 4:30-5:50pm starting Jan 18 2016
- Enrollment – there should be room for everyone meeting the prereq
  - Classroom holds 132. There are 99 registered. 15719 will have <= 87; 18847b will have <= 45.
  - The waitlist has 66 names. Many do not have the prereq.
  - Historically 25% drop when first project is due. That will be Jan 25. If we find another TA, everyone with the prereq should get in. If not, everyone with the prereq might still get in.

Projects …

15-719/18-847b
Advanced Cloud Computing

Lecture 01

Projects and Infrastructure

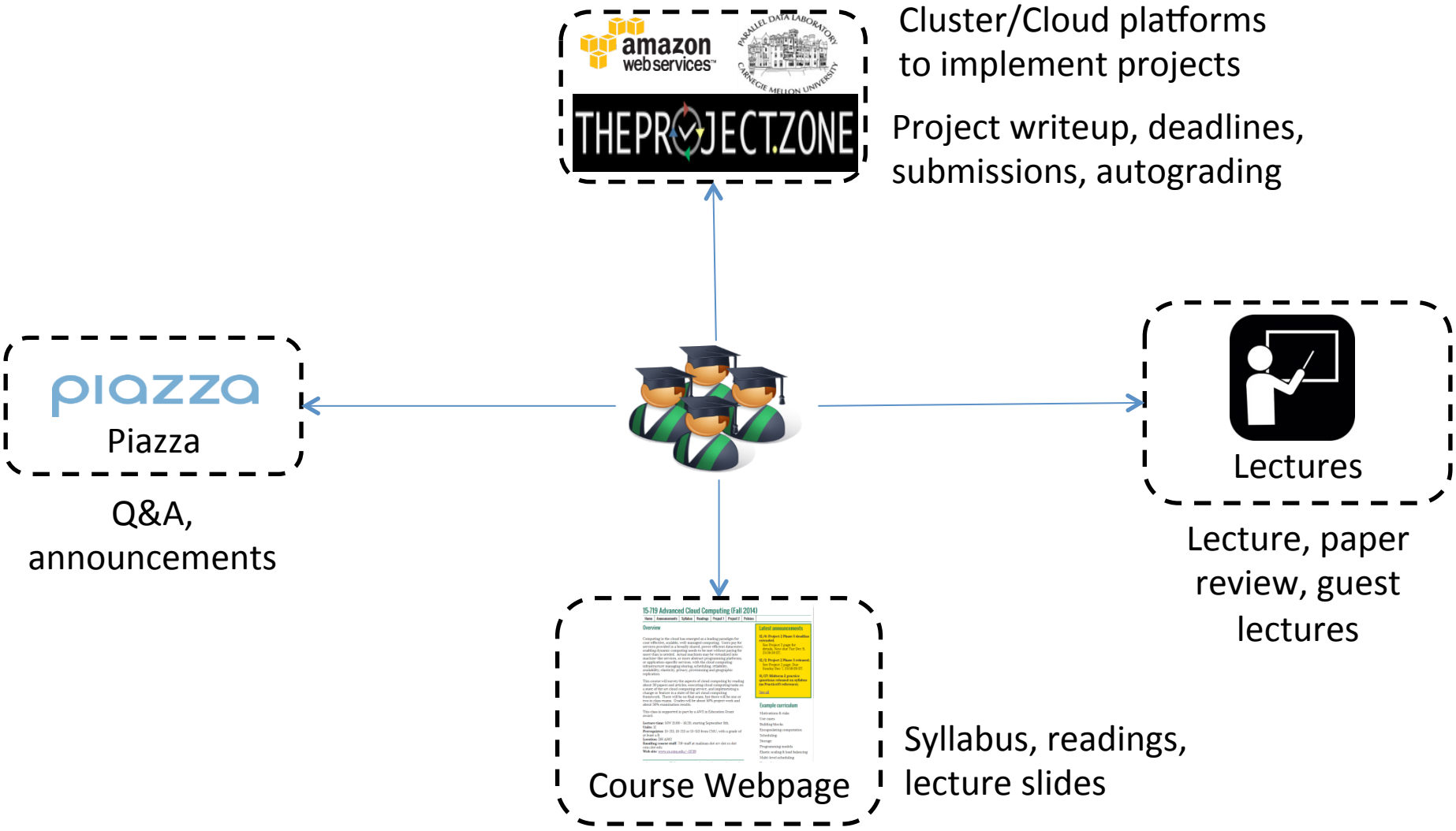
January 18, 2017

http://www.cs.cmu.edu/~15719/

Projects

- Three Individual Projects:
  1. Scalability and Elasticity (Out 1/18, Due 1/25)
     - AWS
  2. Programming Frameworks (~ 4 weeks)
     - Spark
  3. Building and Scheduling Clouds (~ 4 weeks)
     - YARN

Course Engagement Model



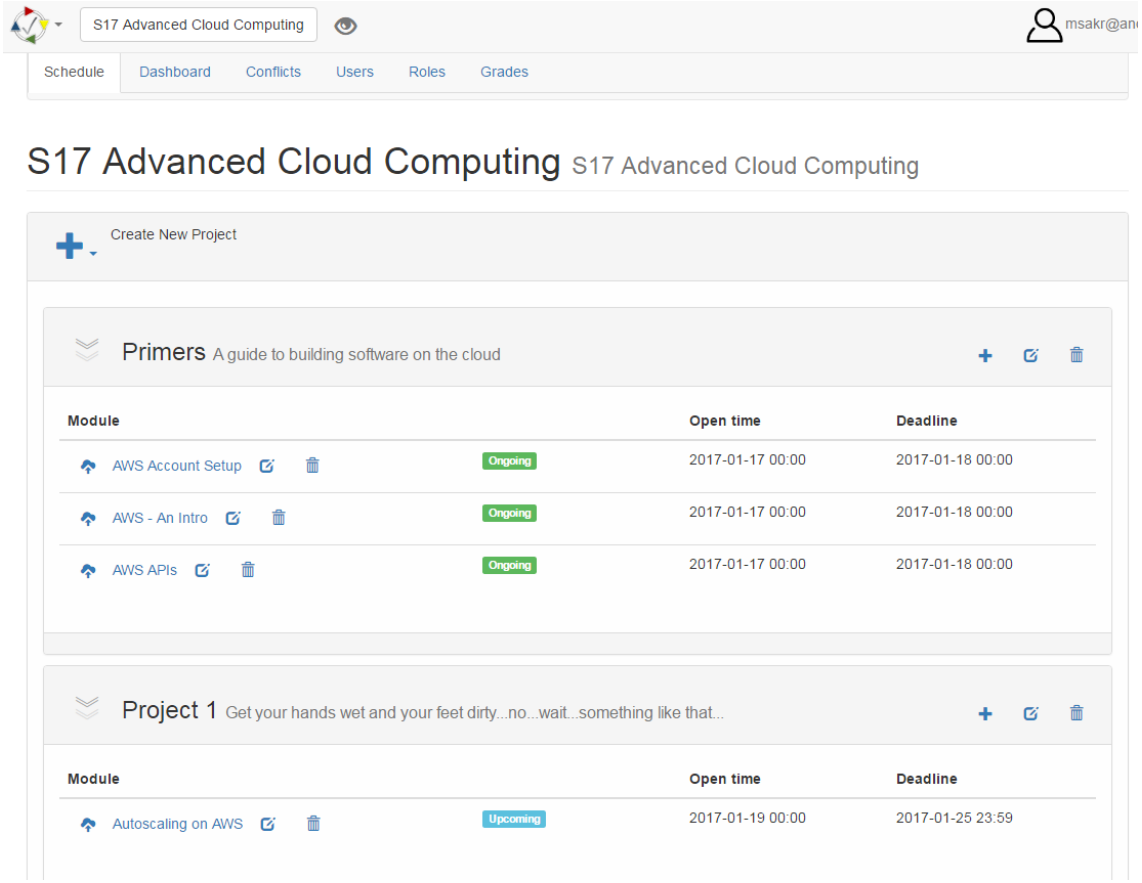
Course Administration

- A *single* Piazza course page is created
  - https://piazza.com/cmu/spring2017/15719/home
  - Enrolled students will be invited to Piazza
- Schedule of projects is on TheProject.Zone
  - Project deadlines are posted on
    - https://TheProject.Zone
- Course webpage will be updated soon
  - http://www.cs.cmu.edu/~15719/

TheProject.Zone

Course projects are on https://TheProject.Zone:

- Learn through repetitive attempts and feedback
- Enrolled students are automatically registered
- Access through browser
  - Not mobile friendly yet
- Work in progress
  - We will encounter bugs
  - Provide feedback on Piazza
  - Please be patient



Tentative Schedule

- Assessments:
  - 2 in-class exams
  - 3 projects on TheProject.Zone

Week	Week Starting Monday	Assessments	Date
1	1/16/2017	P1 Out	1/18/2017
2	1/23/2017	P1 Due	1/25/2017
3	1/30/2017	P2 Out	1/30/2017
4	2/6/2017		
5	2/13/2017		
6	2/20/2017		
7	2/27/2017	P2 Due	2/27/2017
8	3/6/2017	Exam 1	3/8/2017, 6-9pm
9	3/13/2017	Spring Break	
10	3/20/2017	P3 Out	3/20/2017
11	3/27/2017		
12	4/3/2017		
13	4/10/2017		
14	4/17/2017	P3 Due	4/17/2017
15	4/24/2017		
16	5/1/2017	Exam 2	5/3/2017, 6-9pm

Tentative Grading

Course Elements	#	Weight
Projects	3	50%
In-class Exams	2	50%

Academic Integrity

It is the responsibility of each student to produce her/his own original academic work

- All projects in this course will be individual work

Read the university policy on Academic Integrity

The Penalties are Severe

- Cheating leads to several students being dismissed from the university every semester

LET IT NOT BE YOU!

What is Cheating

- Sharing code or other electronic files either by copying, retyping, looking at, or supplying a copy of any file.
  - Other students, github, stackoverflow, anywhere on the internet,…
- Copying answers to any assessment from another individual, published or unpublished written sources, and electronic sources.
- Collaborating with another student or another individual on exams or projects.
- Sharing written work, looking at, copying, or supplying work from another individual, published or unpublished written sources, and electronic sources.
- …(read the university policy)

Special Note on Amazon EC2

- AWS has generously donated resources for this course
- Paid Cloud Service – billed by the hour
- Start a resource only when you need it
- To explore, use a micro instance
  - You can keep one micro instance running 24x7
- Terminate all other resources as soon as you are done with them
- Students will be penalized for careless usage
  - You will be provided with a $50 coupon to use on AWS
  - The coupons will be distributed via theproject.zone

This Week

- Projects on TheProject.Zone
  - P1, due Wednesday, Jan 25, 2017
- Check that you are enrolled on Piazza
  - https://piazza.com/cmu/spring2017/15719/home
  - If not enrolled, email 719-staff@mailman.srv.cs.cmu.edu
- Complete Primers on theproject.zone
  - Create an account on AWS (ASAP)
  - Submit your AWS account info using your profile on TheProject.Zone
  - Redeem the $50 coupon provided

Motivations, definitions, obstacles: readings

- Reference 1: “A View of Cloud Computing,” Armbrust, Fox, Griffith, Joseph, Katz, Konwinski, Lee, Patterson, Rabkin, Stoica, Zaharia, CACM April 2010. http://doi.acm.org/10.1145/1721654.1721672
- Reference 2: “The NIST Definition of Cloud Computing,” Mell, Grance, NIST Special Publication 800-145, Sept 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Optional readings

- Opt. Reference 3: "NIST Cloud Computing Reference Architecture," Liu, Tong, Mao, Bohn, Messina, Badger, Leaf, NIST Special Publication 500-292, Sept 2011. http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST_SP_500-292_-_090611.pdf
- Opt. Reference 4: "Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS," Rackspace Support. September 2012. http://www.rackspace.com/knowledge_center/whitepaper/understanding-the-cloud-computing-stack-saas-paas-iaas
- Opt Reference 5: "PaaS is not middleware of IaaS," Reza Shaffii, Oracle, Jan 2012. https://blogs.oracle.com/rezashafii/entry/paas_is_not_middleware_over
- Opt Reference 6: "The Rise of Cloud Computing Systems," SOSP History Day, SOSP 2015. http://dl.acm.org/citation.cfm?id=2830903.2830913

Historically speaking

- First computers were dedicated to a few apps, running one at a time
- Then users shared same big computer interactively (timesharing)
- Then PCs got powerful & cheap enough to dedicate to single users
- Then users shared a cluster of nodes, running one job at a time
- Now we are working on sharing same clusters of computers interactively (cloudsharing?)

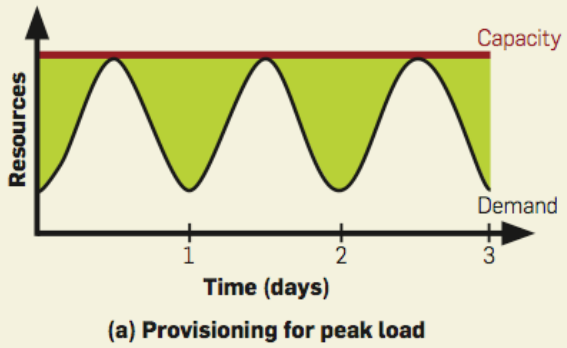
- Meanwhile networking started as dialing up the server once in a while, then it became a revenue driver (justifying building out costly infrastructure), and now society assumes electronic access at all times

Multi-tenant consolidation with elastic cost conservation

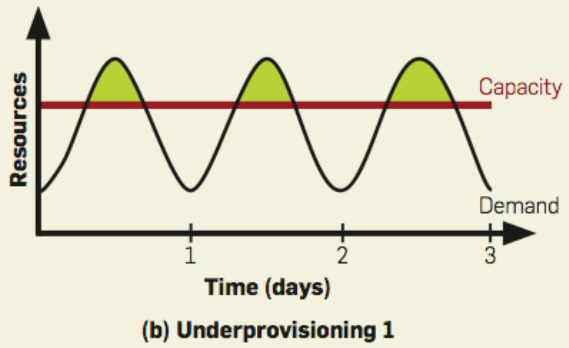
- Properties of cloud computing:
  - Computing as a utility – always available, basics (power, water, phone)
    - Computing is across the network, accessible by all types of devices
  - Simplified interface/management – use the right API for user needs
  - Share resources over different users/uses – statistical multiplexing
  - Economies of scale from consolidation – better amortized costs
  - Convert capital costs to operating costs – short leases instead of buying
  - Rapid and easy variation of usage – plug in more load on-demand
    - Appearance of infinite resources – all users are small users
  - Pay only for what you use – fine grain metered usage determines the bill
  - Cost conservation: 1 unit for 1000 hours == 1000 units for 1 hour

Consolidation, sharing, elasticity

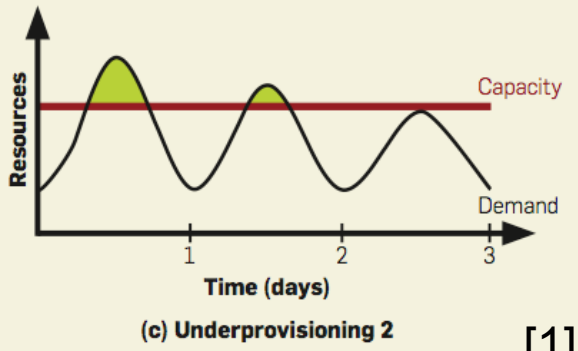
- Statistics tells us that average of i.i.d. variables has a decreasing variance independent of the variables' distributions (CLT)
  - Lots of users with widely varying needs apply a considerably less variable load on a huge provider – providers can manage overprovisioning well
  - Users perceive exactly what they need all the time, if their needs are small



(a) Provisioning for peak load



(b) Underprovisioning 1



(c) Underprovisioning 2

[1]

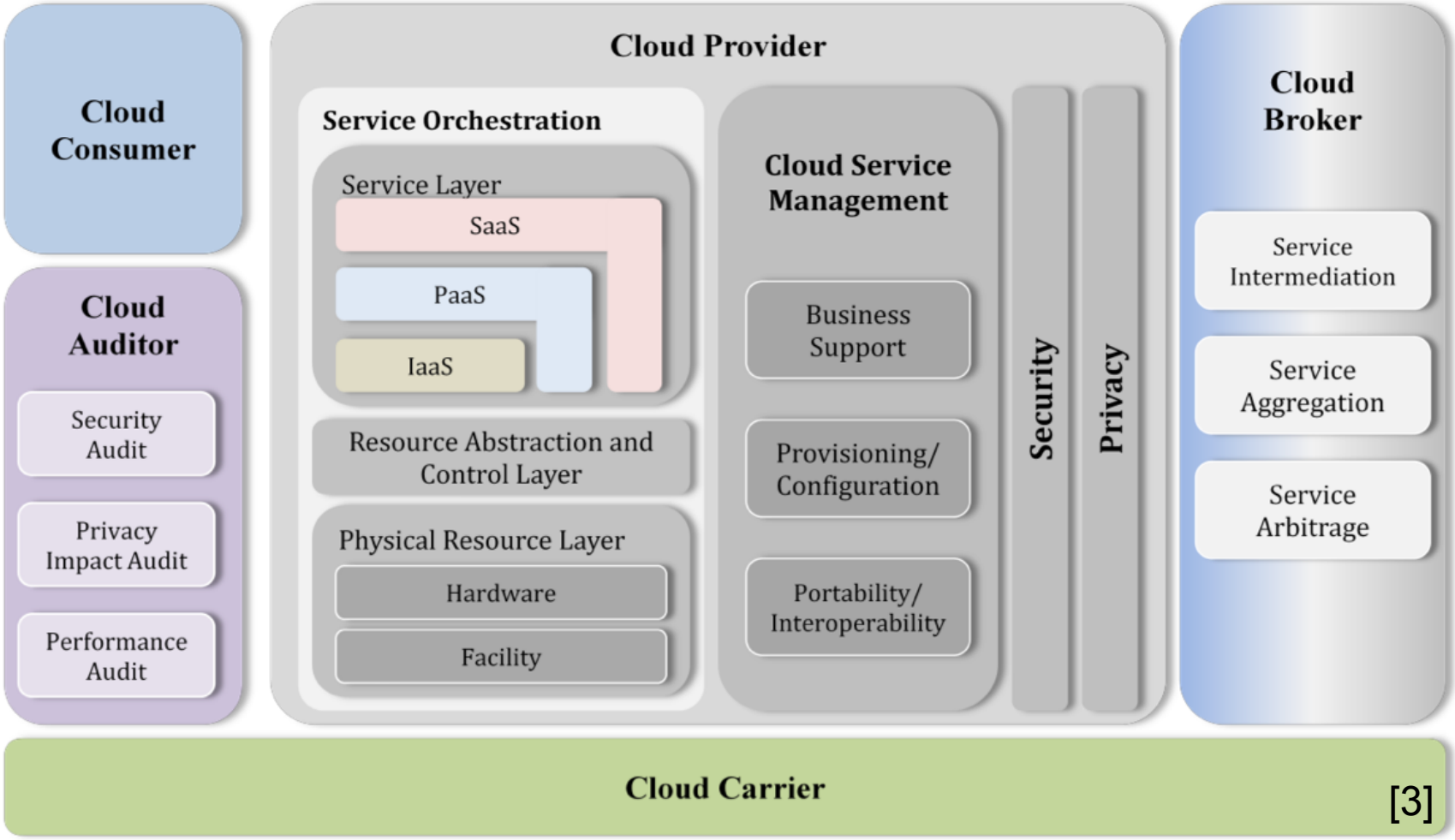
The right API – SaaS, PaaS, IaaS etc

- A (flawed) taxonomy is Service, Platform or Infrastructure as a Service
  - SaaS: service is a complete application (client-server computing)
  - PaaS: high level (language) programming model for cloud computer
    - Eg. Rapid prototyping languages
    - Turing complete but resource management hidden
  - IaaS: low level (language) computing model for cloud computer
    - Eg. Assembler as a language
    - Basic hardware model with all (virtual) resources exposed
- Higher level abstractions simplify matching user tasks and free users from details of provisioning, configuration, fault tolerance etc
- Lower level abstractions enable apps to achieve user-specific goals
- Eg. Salesforce (SaaS), Google AppEngine (PaaS), Amazon AWS (Iaas)

Deployment models

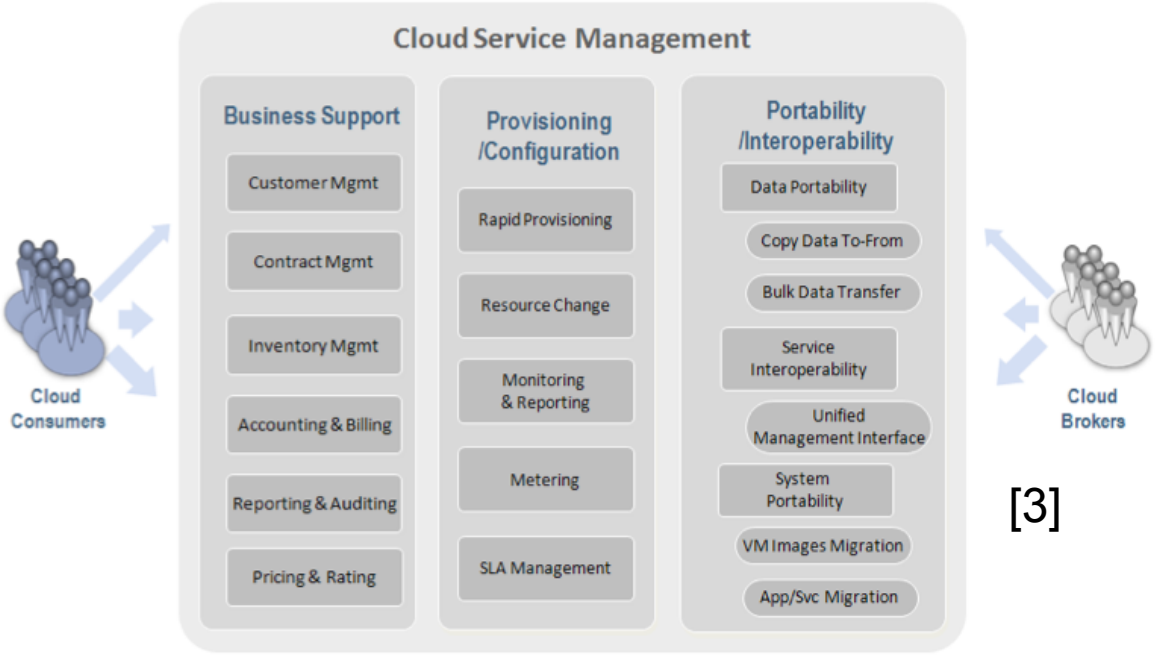
- Real ownership defines deployment models
  - Public cloud – provider sells computing to many unrelated consumers
  - Private cloud – provider is one organization with many largely unrelated components and divisions as consumers (may also be outsourced)
  - Community cloud – providers and consumers are different organizations with strong shared concerns (federations)
  - Hybrid cloud – Multiple providers combined by same consumer
    - Eg., Better availability, overflow from private to public, load balancing to increase elasticity

NIST Architecture



[3]

NIST Arch Con't



[3]

- Actors are roles
  - Auditors should be independent of providers
  - Carriers are network providers
  - Brokers mix and match consumers to providers
    - May "Value-add" e.g. identity mapping, quota enforcement
    - May "Mashup" different services into a new service
    - May "Arbitrage" pick the best based on value for money
  - Broker Eg.: DuraCloud for cloud storage;
    CloudSwitch for a private PaaS impl'd in cloud
- Provider support for business, resources, portability
  - Making it easier for consumer to use cloud computing

Larry Ellison's objection

- "defined cloud computing to include everything we already do"
- Ie., timesharing: many users slice up a single big computer
  - Virtualize the hardware as protected processes with shared storage
  - Accumulate usage accounting for billing/cost-sharing
- Ie., client-server computing: program API for communication
  - Relieve clients from burden of operating, provisioning server
  - Set abstraction of server function at a natural/constrained level
- Ie., leasing companies – spread the cost over computer lifetime
  - Rates depend on commitment: 100% -> prime rate loan,
    lower commitment to pay leads to high "interest" rates

Client-Service evolution (SaaS)

- First there was RPC (remote procedure call)
  - Simple modularity of code in two different failure domains
- Which became DCE/DCOM/CORBA client-server computing
  - Set of language free services that modular code uses to
- Which became .NET/SOAP
  - Service oriented architecture is principled modularity & interoperability
- Which became REST/RMI/Thrift
  - The tools used to reach Software as a Service today

Obstacles to cloud computing use

- Privacy & security
  - How is this different from outsourcing payroll, legal, HR, email?
  - Its new so standard practice, certifications, insurance etc not in place
  - Its often business critical/proprietary (topline revenue providing)
  - Uses code running aside other users opens unknown attack channels
- Privacy in the world tends to rely on regulation
  - we know encryption is too weak for safety from very big organizations
  - will cloud also need to rely on regulation for privacy?

Obstacles to cloud computing use

- Utility issues (power, water, phone, ….)
  - Need very high availability, must be able to get your data out, who is liable for failures?
  - Little consumers have little leverage on monopolistic providers
  - Playing multiple providers off against each other may help, standards too
  - Can you trust your cloud provider? Is your Google-hosted email safe from Google lawyers when you attempt to sue Google for breach of contract?
- Physical utilities tend to rely on regulation
  - utilities must meet government standards of service for consumers
  - will cloud also need to rely on regulation?

Obstacles con't

- High cost of networking combined with always remote
  - Telephone/networking moving to per-GB charges (& unlimited minutes)
    - $10/GB over cell (movie rental == download charge)
    - $0.10/GB over internet (disk price == download charge)
  - With these charges, why move data to compute?
    - AWS offers free bandwidth to data stored and computed in AWS
    - Regional centers allowing customer "dark fiber" access

Obstacles con't

- Performance unpredictability & in situ development/debugging
  - Virtual resources much less tangible than physical resources
    - VM performance depends on core locality, memory availability, network congestion, disk congestion, etc – makes scheduling and debugging harder
    - Scaling implementations are hard – elastic applications, storage, low latency
  - Bugs often appear only at scale, in the situation of use (in situ)
    - Debugging facilities best for provider, worst for consumer
  - Consumer reputation shared with other consumers – need trusted auditor
- Software licensing – $/yr/CPU is not elastic and pay as you go

Marketeering extensions (XXX as a Service)

- Data as a Service (DaaS)
  - Collector & seller of useful data
  - Analytics on consumer data, eg., Aciom (www.aboutthedata.com)
- Network as a Service
  - Value-added service improving your data network
  - Content Delivery Networks (CDN) eg. Akamai
- Communication as a Service (CaaS)
  - A cloud based value-added switching service
  - No-hardware private VoIP switching eg., PBX/IP-Centrex
- IT as a Service (ITaaS)
  - When the IT group in a company competes for the business of a division, instead of being mandated by corporate leadership (profit, not a cost center)
- XXX as a Service, today just means doing business through a web browser app

Next day plan

- Start Project 1 asap!
- Read papers assigned
  - You may be asked questions about the readings in any lecture
- Prepare to discuss examples of cloud computing
- If you are on the waitlist and do not have the prereq…
  - get your academic advisor to send email to Majd, Greg and Garth explaining why you are prepared for this course – you might get in, maybe

15-719/18-847b
Advanced Cloud Computing

Lecture 01

Project 1 Overview

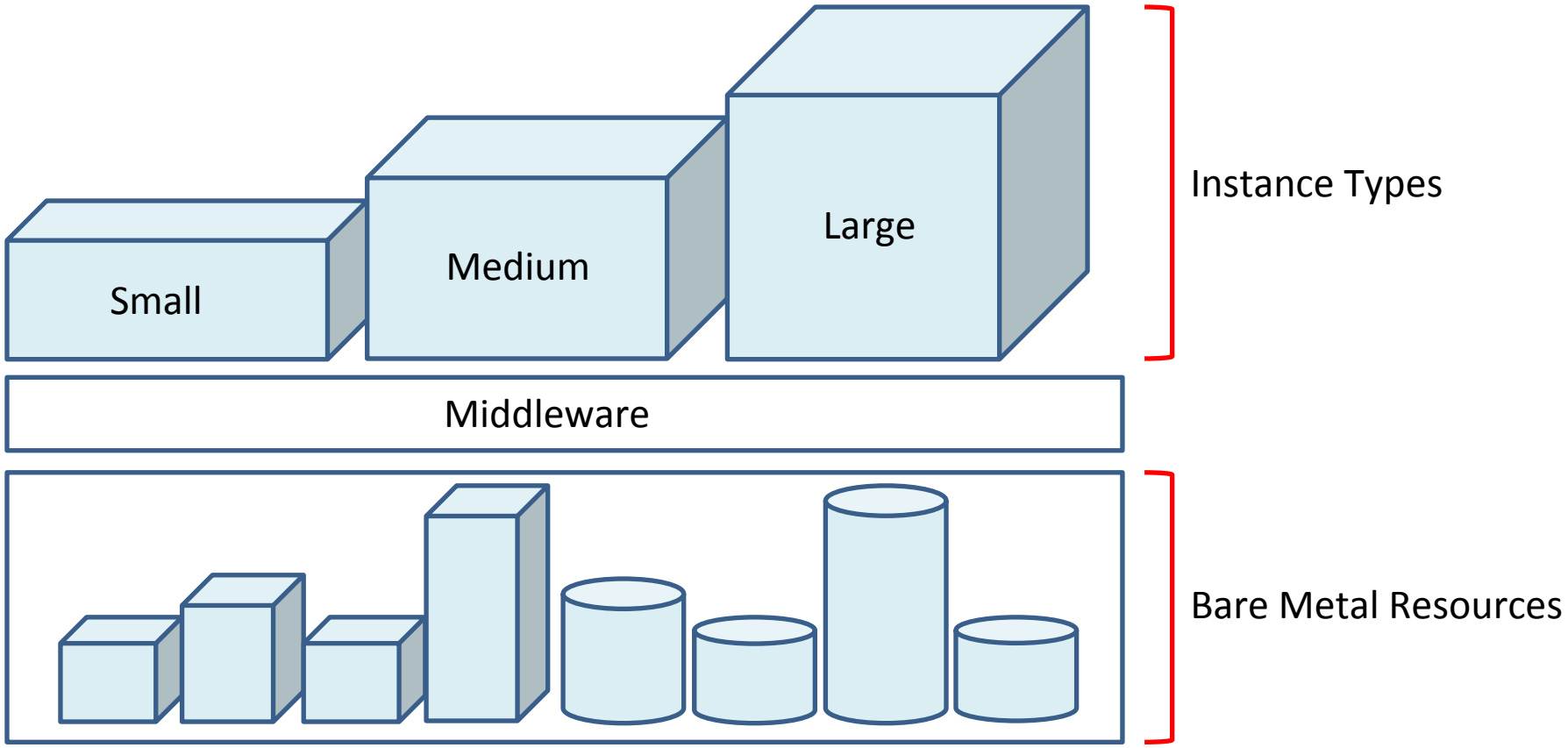
January 18, 2017

http://www.cs.cmu.edu/~15719/

Project 1: Scalability and Elasticity

- Due 1/25/2017
- Learning Objectives
  a. Write code that invokes AWS APIs to horizontally scale VMs in response to demand.
  b. Deploy web services that account for failure, cost and performance constraints.
  c. Identify and explain the need for handling resource failures.
  d. Explain the need for distributing load evenly among all resources.
  e. Configure and deploy an Elastic Load Balancer along with an Auto Scaling Group on AWS.
  f. Develop solutions that manage cloud resources with the ability to deal with resource failure.
  g. Account for cost as a constraint when provisioning cloud resources.
  h. Analyze the trade offs and cost implications of maximizing performance over time.

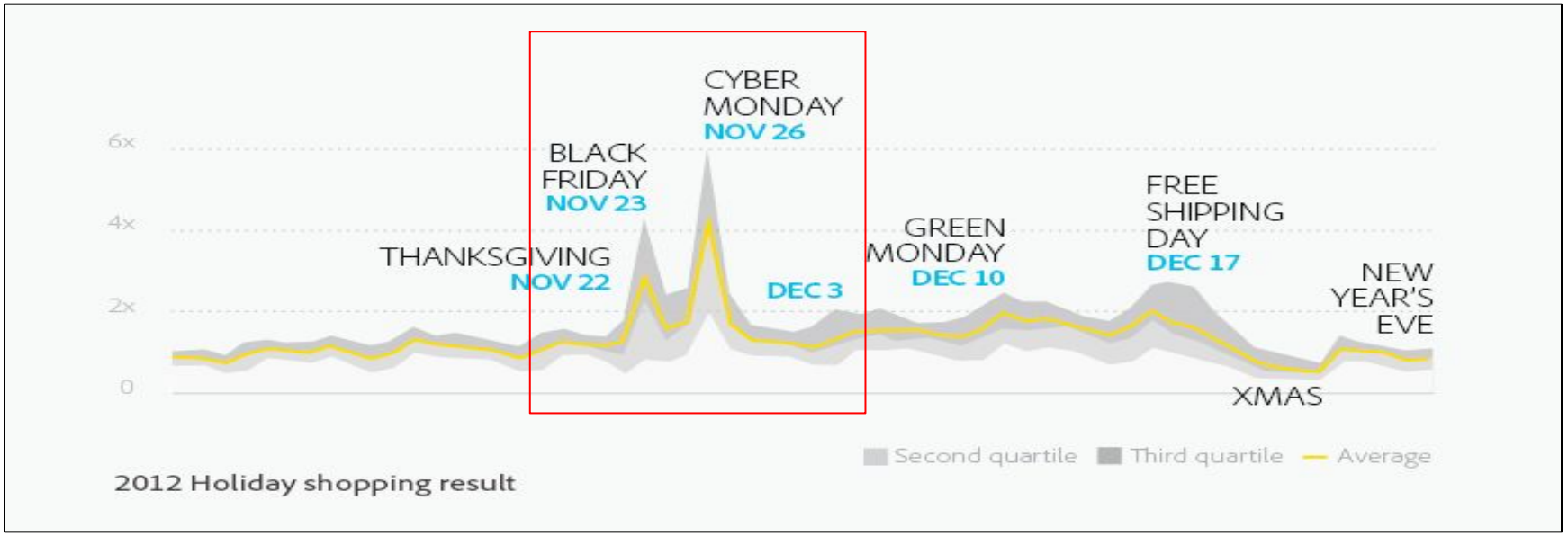
Resources in Cloud Infrastructure



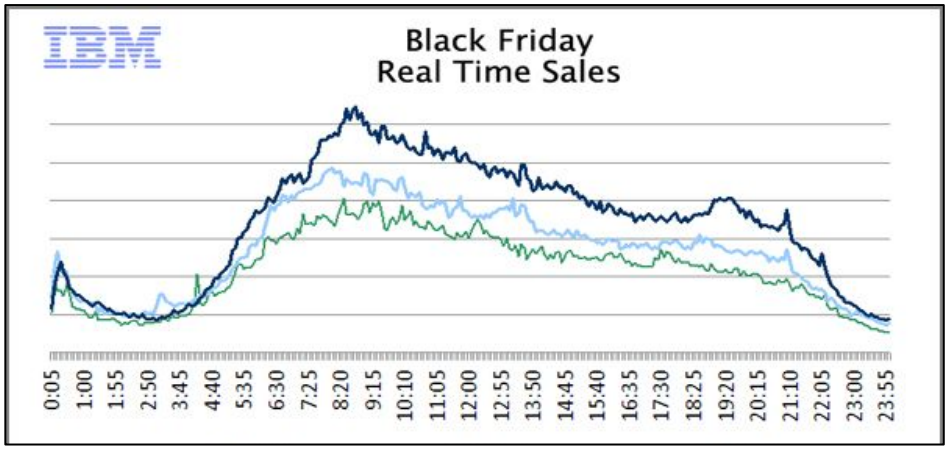
Amazon APIs

- Supported APIs
  - Command Line Interface API Tools (link)
  - AWS SDK for Java (link)
  - AWS SDK for Python (link)

Auto Scaling

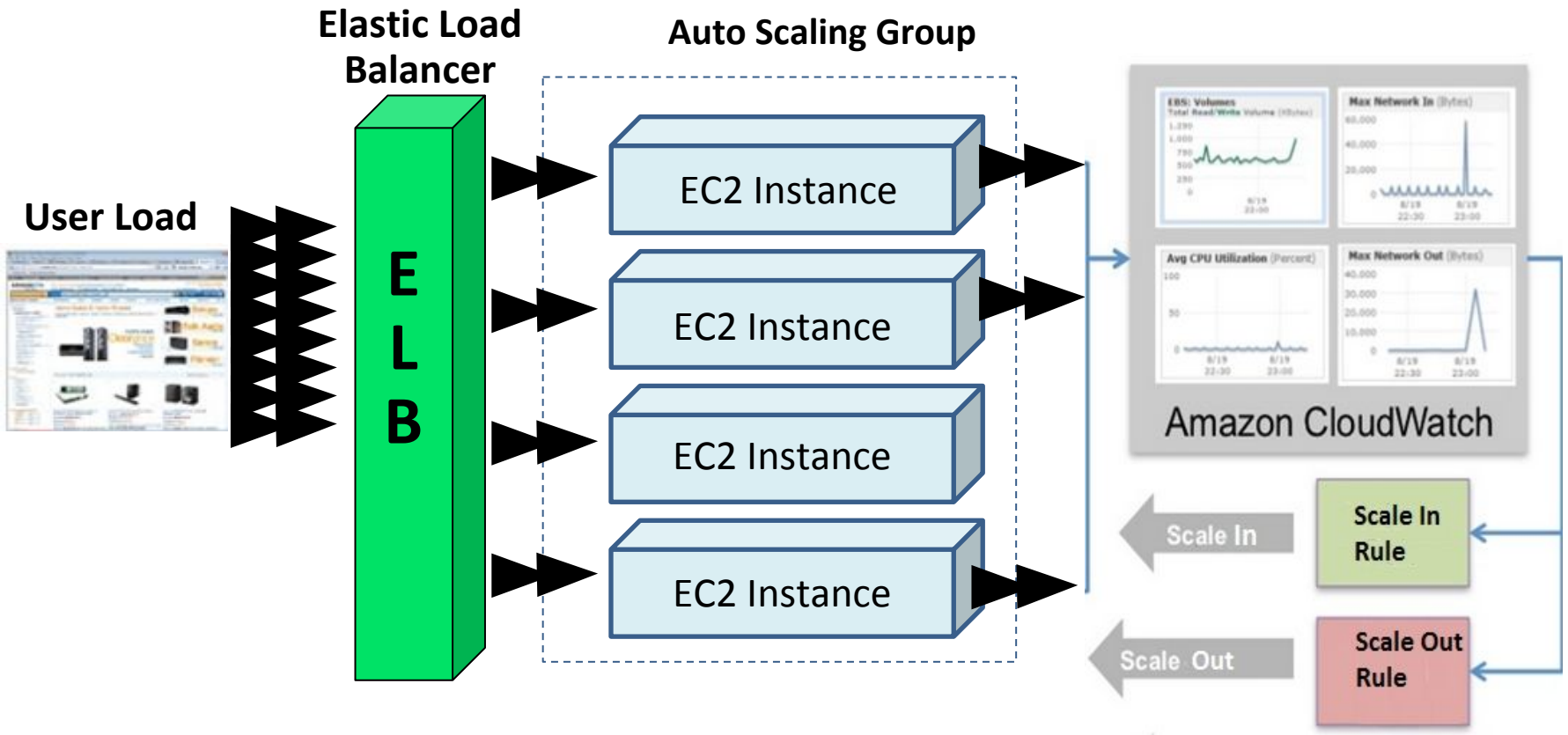


sapient.com



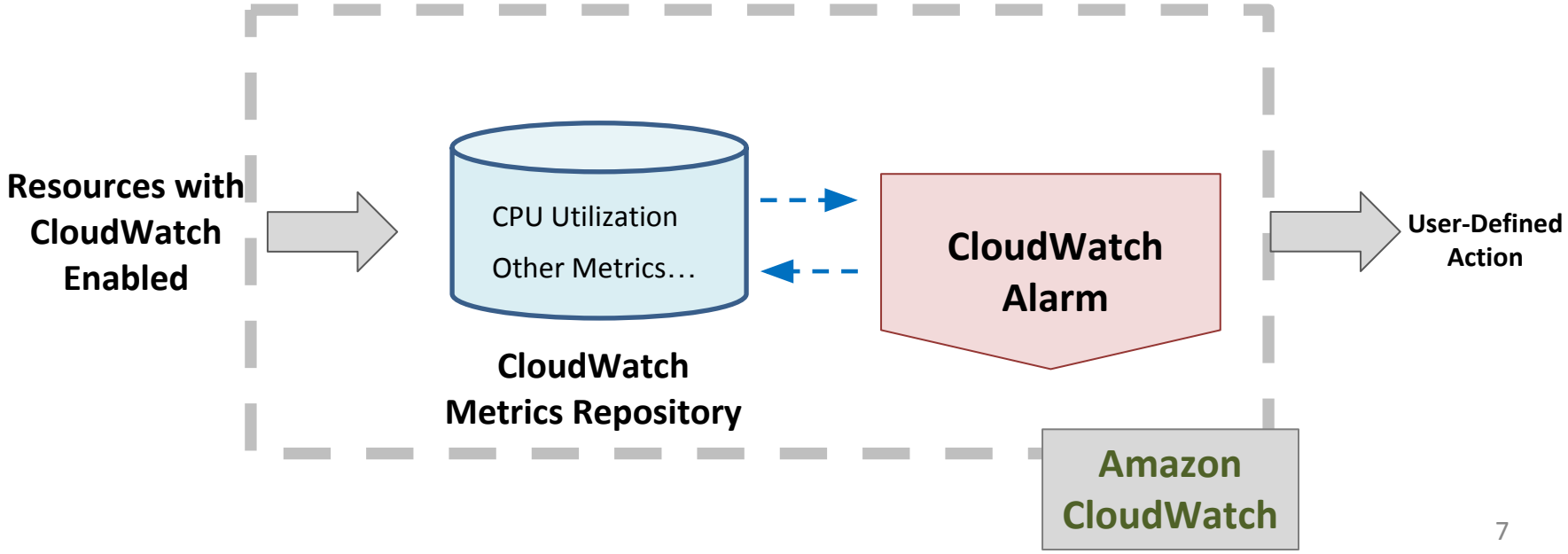
- Burst during the holiday season
- Losing customers with poor service
- Should the size vary with traffic?

P1 Architecture
Amazon Auto Scaling Group



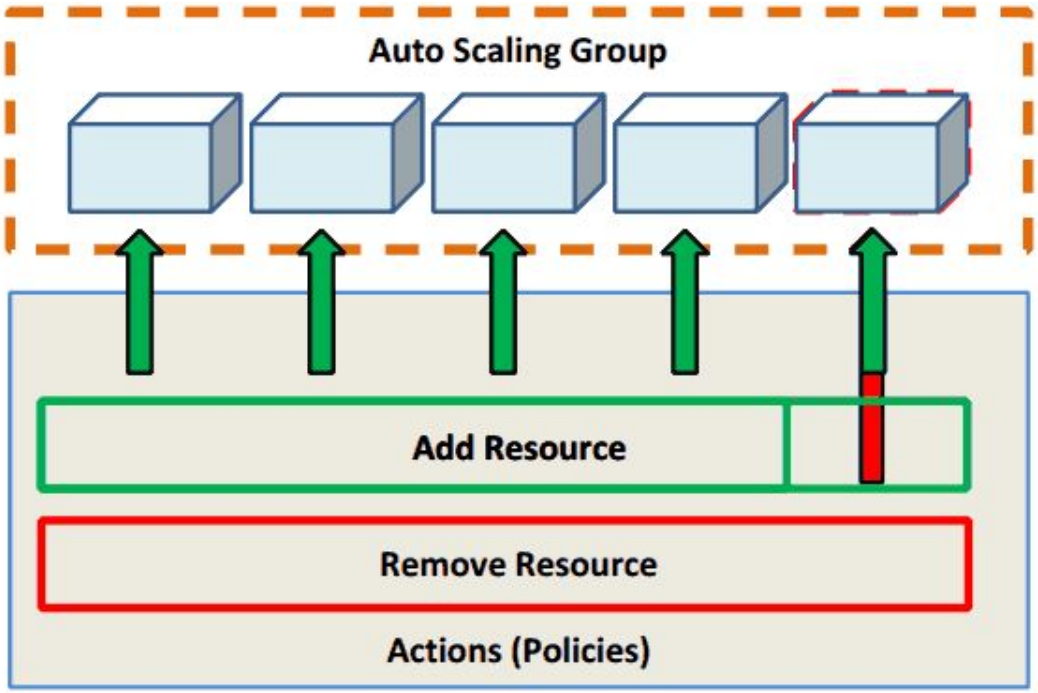
Amazon's CloudWatch Alarm

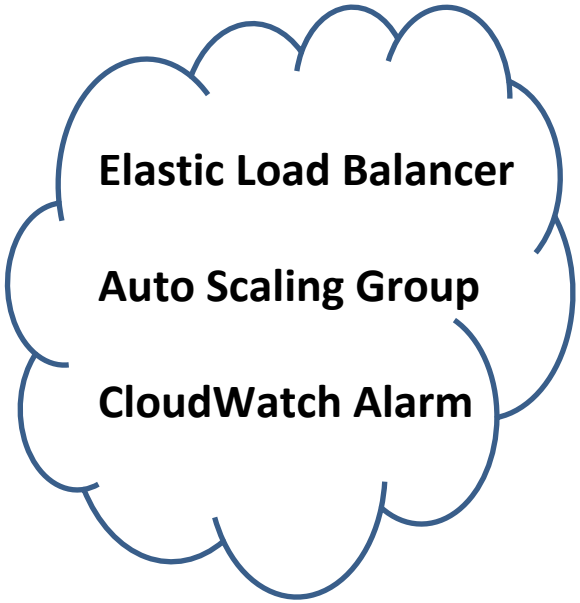
- Monitor CloudWatch metrics for some specified alarm conditions
- Take automated action when the condition is met



P1 - Your Task

- Programmatically create an Elastic Load Balancer (ELB) and an Auto-Scaling Group (ASG) linked to ELB
- Test by submitting a URL request and observe changes
- Decide on Scale-Out and Scale-In policy
- Mitigate failure





Hints

- Use spot instances while practicing, but...
- … don’t use spot instances for your final submission
- Naive scaling based on CPU Utilization is not smart
- CloudWatch runs on all your AWS resources, which metrics are the most useful
- Instances may fail; can you spot a pattern? Detect and recover as soon as possible
- What is the trade-off between medium and large instances?
- When making remote calls, rely on timeouts and retries
- Remember, you need not scale in/out one instance at a time

Resources

- Amazon's Auto Scaling Service
  - http://aws.amazon.com/autoscaling/
- Amazon's CloudWatch Alarm
  - http://aws.amazon.com/cloudwatch/
- Amazon's Scaling Developer
  - http://aws.amazon.com/autoscaling/developer-resources/

Questions?