DISTRIBUTED RESEARCH ON EMERGING APPLICATIONS & MACHINES
dream-lab.in | **Indian Institute of Science**, Bangalore

DREAM:Lab

SE252:Lecture 2, Jan 9 *& Jan 16*

ILO2:1:Cloud Virtualization, *Abstractions and Enabling Technologies*

Yogesh Simmhan

©Yogesh Simmhan, 2013
This work is licensed under a Creative Commons Attribution 3.0 Unported License.

Summary of Lecture 1

Facebook: A Canonical Cloud Based Application

- Scale **Up** vs Scale **Out** of resources
  - Meet growing users/content richness using *more* machines rather than *faster* machines
  - Key concept for *distributed* computing on Clouds
- Service Oriented Architecture (SOA)
  - Standards for "Remote Procedure Call" over network, data structure de/serialization
  - Technology enabler, along with *Virtualization*

Facebook…

- Incremental Pagelet loading using BigPipe
  - Pipelining, Data parallel/MR, Task graph
  - Patterns for building cloud applications
- Synchronous vs. Asynchronous IM Chat
  - Push vs. Pull invocation, guarantees
  - Cloud app execution models & coordination
- Data locality using CDN for photos
  - Trade-off latency vs. freshness vs. consistency

Cloud is ~~a revolution~~ **an important evolution**

1. Cloud computing as a technology
2. Cloud computing as a distributed systems environment
3. Cloud computing as a research topic

Pre-requisites

- UG Algorithms/data struct/OS/networks
- Programming
  - Network programming, Java/Python

Intended Learning Outcomes (ILO)

- **What** you need to be *able to do* after *learning* from the course…expectations

Teaching and Learning Activities (TLA)

- Meet ILOs thru (Guest)Lectures, Project, Homework, Research Reading, Exam

Assessment *(1000 point scale)*

- **20%** for two homeworks
- **10%** for research Summary on a paper
- **40%** for 4 project modules w/ *extra credits*
- **25%** for midterm+final exams
- **5%** for participation
- Schedule … See webpage

http://www.serc.iisc.ernet.in/~simmhan/SE252/#section9

Academic Integrity

- Students must uphold IISc's Academic Integrity guidelines
  - Penalties include a failing grade
- Discussions & reference to online material is encouraged, but you cannot collaborate
  - Must complete assignments by yourself
  - Document contribution of each team member
- **Plagiarism** is not acceptable!
  - You must cite external help/sources
- Participate: There are no "stupid" questions☺

http://www.iisc.ernet.in/students-corner/existingstudents-academicintegrity.php

Resources

- ***Distributed and Cloud Computing: From Parallel Processing to the Internet of Things***, Kai Hwang, Jack Dongarra and Geoffrey Fox, *Morgan Kaufmann*, 2011
- **Official Mailing List:** se252.jan14@mailman.serc.iisc.in
  - *Don't use Digest Mode! Can cause delays.*
- **Lecture:** TTh 1130AM-1PM, SERC 202
- **Office Hours:** ~~Wed, 330-5PM~~, Fri 1130-1pm SERC 411

Auditing? Schedule conflict? Etc.

- Final date for joining class is **before** Jan 14th!
  - *NOTE: No class on Jan 14 due to campus holiday*
  - *NOTE: Special session on Wed 15 Jan 1-2PM will be used to discuss Project #0*

Ongoing Assignments

Project Preliminaries

- Form teams of two. Pick cool & unique title ☺
  - ~~Sign up for AWS account using team title~~
  - ~~http://aws.amazon.com/ -> Sign Up~~
- Email me team name & members info to me **before Sunday Jan 12th**
  - *CC your team member in the email*
- Credit students pair up with credit students. Audit w/ audit.

Reading Assignment

- Chapter 1 of text book

Misc Tasks

- Sign up on mailing list using IISc email ID
  - http://mailman.serc.iisc.in/mailman/listinfo/se252.jan14
- Send status update & mugshot

DREAM:Lab

Lecture 2

ILO 2 *(NOTE change in ILO1 & ILO 2 order…)*

- Cloud Virtualization, Abstractions and Enabling Technologies
  - *Explain* virtualization and their role in elastic computing.
  - *Characterize* the distinctions between Infrastructure, Platform and Software as a Service (IaaS, PaaS, SaaS) abstractions, and Public and Private Clouds, and
  - *analyse* their advantages and disadvantages.
  - *Describe* service oriented architectures that are foundational to the WWW.

Say you have a house to rent…

- What does the tenant want?
  - An independent house ☺
- What can you give?

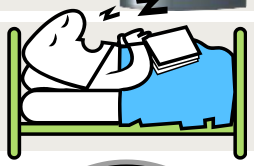
What does a tenant look for?

- Is it affordable?

- Is there enough space?

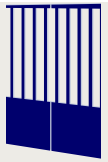
- Is it safe from outsiders?
  - Is it safe from other tenants? Locks, shades, ..

- Will I not be disturbed by tenants?



- Is power billed separately?

- Can I get a separate main entrance?
  - Or at least make sure I don't have to fight crowds?



- Do I have to share the verandah!!?

Say you have a computer to rent…

- What does the “tenant” want?
  - Their own computer ☺
- What can you give?
  - And how?

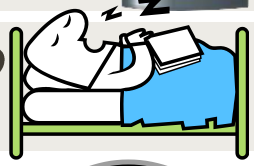
What does a tenant look for?

- Is it affordable to rent?

- Is there enough CPU/memory?

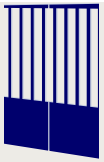
- Is it safe from the N/W?
  - Is it safe from other users? Mem/Code Leaks..

- Will their application use affect me?



- Can I pay for what *I* use?

- Can I get my own N/W connection?
  - Or at least have a reserved bandwidth?



- What do you mean I share the disk!!?

Why Virtualize?

- Share same hardware among independent users
  - Degrees of HW parallelism increasing
- Reduce HW foot print thru' consolidation
  - Eases management, energy usage
- Sandbox/migrate applications
  - Flexible allocation, utilization
- Decouple applications from underlying HW
  - Allows HW upgrades without impact on OS image

Virtualization raises the Abstraction

- Similar to *Virtual Memory* to access larger address space
  - *Physical memory* mapping is hidden by OS using *paging*
- Similar to hardware emulators
  - Allow code on one arch to run on a different
- Physical devices -> Virtual Devices
  - CPU, Memory, VHD, NIC
- Now worry/not be aware of physical hardware details

Virtualization Requirements*

- Efficiency Property
  - *All innocuous instructions are executed by hardware*
- Resource Control Property
  - *It must be impossible for programs to directly affect system resources*
- Equivalence Property
  - *A program with a VMM performs in a manner indistinguishable from another without*
  - *Except: (1) Timing, (2) Resource Availability*

* Formal Requirements for Virtualizable 3rd Generation Architectures, Popek & Goldberg, *CACM*, 1974

Types of Virtualization

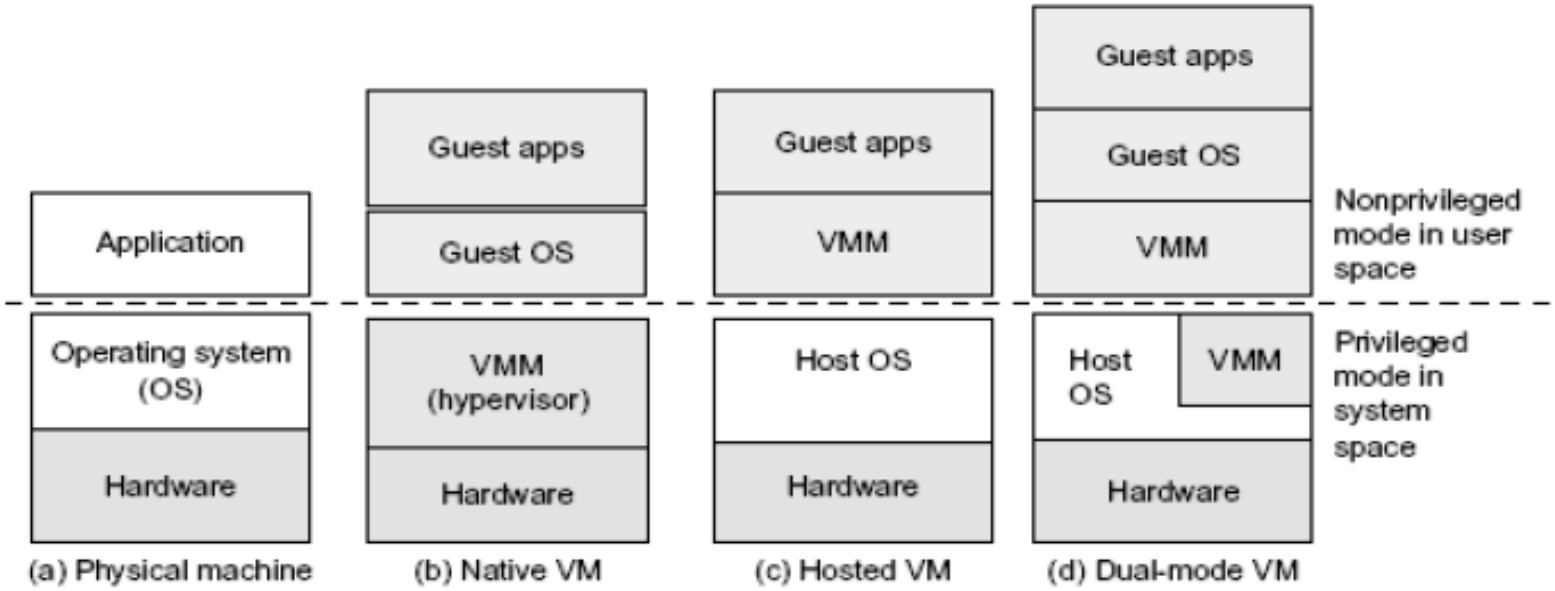


FIGURE 1.12

Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

- Virtual Machine Manager/Virtual Machine Monitor/Hypervisor … *a Caretaker*
- Native (Hyper-V, ~KVM), Hosted (Xen)

Fig. from "Distributed & Cloud Computing, Hwang, et al, MKP; *Courtesy of Kai Hwang, USC*

Types of Virtualization*

- Full Virtualization
  - *Unmodified* Guest OS
  - VMM *binary translates kernel* to *trap* privileged calls
  - Software emulation
  - VMWare Server, Apple Parallels
- Pros
  - Guest OS not modified
  - No HW support required
- Cons
  - Binary translation costly, difficult

?

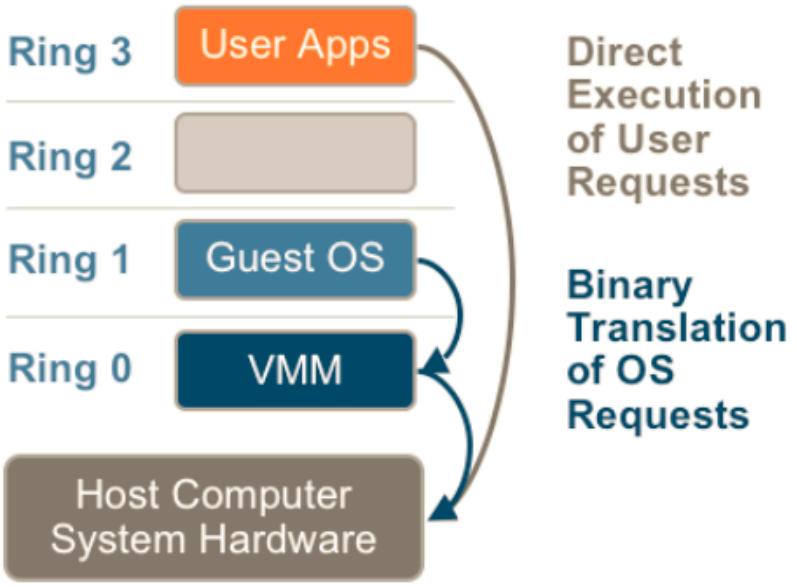


Figure 5 – The binary translation approach to x86 virtualization

*Understanding Full Virtualization, Paravirtualization and Hardware Assist, VMWare, Tech Report WP-028-PRD-01-01, 2007

Types of Virtualization

- Para-virtualization
  - Guest OS *modified* to make "hyper-calls" for privileged instructions
  - Xen in para mode
- Pro
  - (Mostly) faster & easier than bin. translation
- Con
  - Guest OS modified… Legacy, maintenance

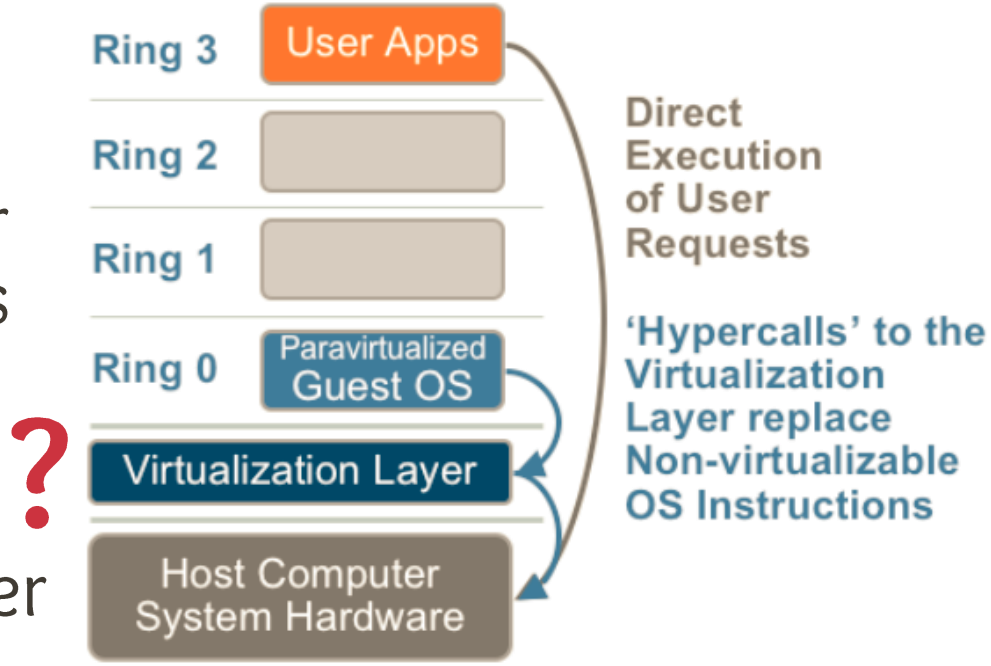


Figure 6 – The Paravirtualization approach to x86 Virtualization

Types of Virtualization

- H/W Assisted Virtualization
  - *Unmodified* Guest OS
  - CPU traps & calls VMM for privileged calls
  - CPU support in Intel VTx, AMD-V
  - Xen HVM, Hyper-V, KVM
- Pros
  - Faster to execute
  - Easier management
- Cons
  - Requires CPU support

?

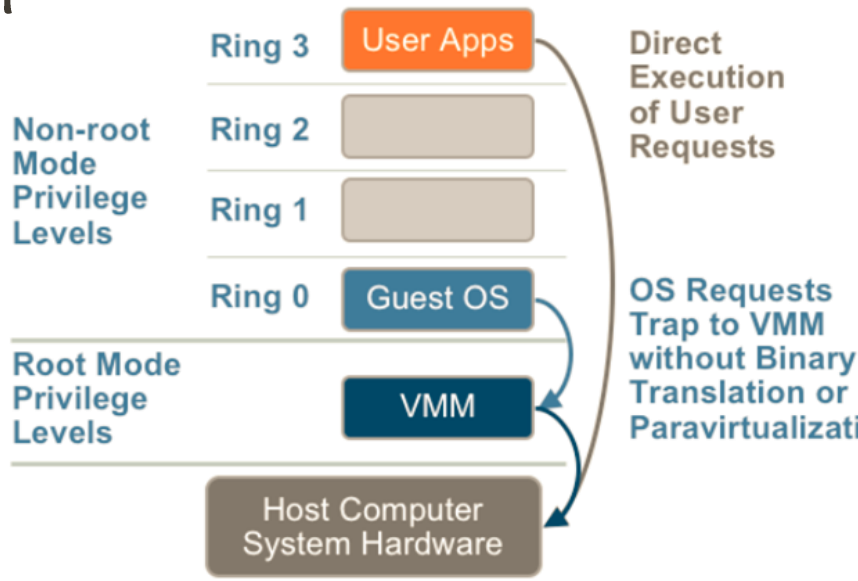


Figure 7 – The hardware assist approach to x86 virtualization

Types of Virtualization

- OS Level Virtualization
  - OS provides containers for isolation
  - Retains host OS image
  - Linux `chroot`, OpenVZ
- Pros
  - Faster to boot
  - Fewer images to maintain
  - No CPU support required
- Cons
  - No guest OS, limited distros

?

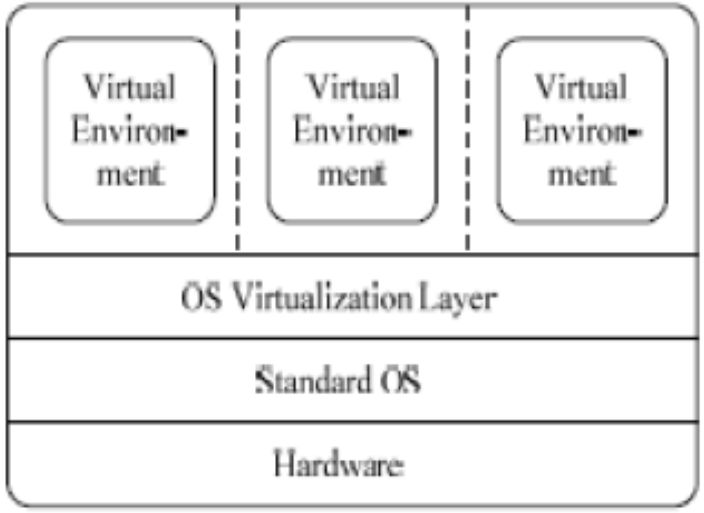


Fig. from "Distributed & Cloud Computing, Hwang, et al, MKP; *Courtesy of Kai Hwang, USC*

Centrality comes a full circle

- Mainframes -> Personal Computers -> Independent Servers -> Enterprise Servers -> Data Centres
- Data centres (More in Lecture #5)
  - Consolidate hardware, infrastructure, energy usage
  - Ease management, automation, physical security
  - Allow transparent HW improvements
- Started as enterprise-scale data centres…

Cisco's Data Center in Texas

Google’s Data Center in Georgia

Microsoft's Data Center in Ireland

NSA’s Data Center in Utah

How does this all relate to Cloud Computing?

- Rent out spare capacity in Enterprise Data Centres
  - Amazon AWS, etc.
- Build Data Centres where HW can be outsourced
  - Rackspace, etc.

A Colony to rent

Levels of Abstractions: IaaS, SaaS, PaaS *(More in Lecture #3)*

- Infrastructure as a Service (IaaS)
  - Rent out virtual machines
  - E.g. Amazon AWS's Elastic Computing Cloud
- Platform as a Service (PaaS)
  - Programming APIs that can be scaled
  - E.g. Microsoft Azure's Workers, Google App Engine, AWS Elastic MapReduce
- Software as a Service (SaaS)
  - End use applications that can be composed & scaled
  - E.g. GMail, Office365, DropBox

And Facebook?

5min Peer Discussion

Reading from Today's Lecture

- Textbook, Sec 3.0 – 3.3
- Understanding Full Virtualization, Paravirtualization and Hardware Assist, *VMWare, Tech Report WP-028-PRD-01-01*, 2007
- Ask questions in next class

Project 0

Not Graded!

Course Project Overview

Book Recommendation using Text Analytics

Browser

Reading History

- *Atlas Shrugged*
- *War & Peace*
- *My Experiments with Truth*
- …

Extract signature

Available Books

- …
- …

Extract signature

Find Similarity & Classify

Make Recommendation

If you like this, you may also like …

Text Analysis Pipeline #1

Call me Ishmael. Some years ago-never ...

Get Book as a Text file → **Tokenize** to Extract words → Remove **Stop Words** → Do **Word Count** → Find Top "N" Words → Build **Word Cloud**

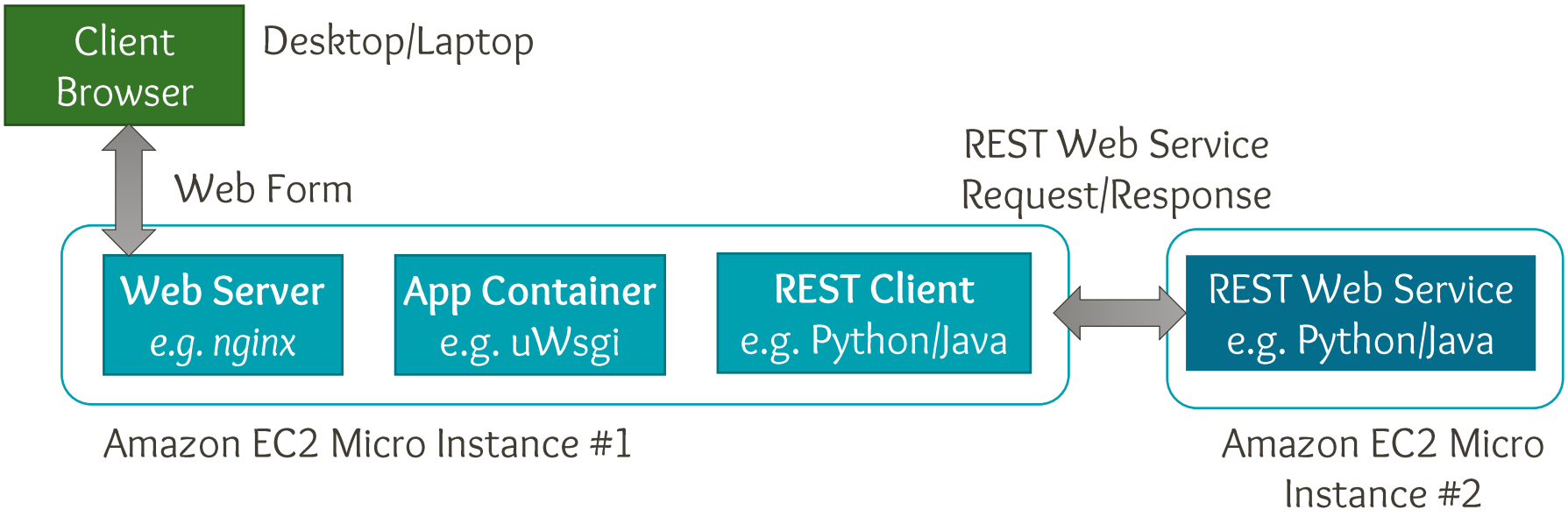
Pipeline #2 will expand on this

Project #0

1. Get team account on Amazon AWS from Yogesh
2. Start Linux micro-sized VM instance on AWS
3. Install & run web server on one micro-instance
4. Write a REST Program to run on another micro-instance
   - GET echo?**msg**=*Hello%20World*
   - Response should just be the **msg** text that was passed
5. Write a web form taking the input **msg** as a text box
   - Call "GET echo" on server side
   - Display the result from the GET request on the webpage

More details will be posted online by Sunday

Setup


Client Browser
Desktop/Laptop
Web Form
REST Web Service Request/Response
Web Server e.g. nginx
App Container e.g. uWsgi
REST Client e.g. Python/Java
REST Web Service e.g. Python/Java
Amazon EC2 Micro Instance #1
Amazon EC2 Micro Instance #2

Project #0

- Pick up info on REST web services
  - Search online!
- You can use **nginx** web server
  - http://kmil.us/blog/2012/11/24/build-a-webserver-on-amazon-ec2/
- For Python, you can use **Flask** for writing REST service & uWsgi to call python from nginx
  - https://github.com/d5/docs/wiki/Installing-Flask-on-Amazon-EC2
  - https://flask-restless.readthedocs.org
- For Java, you can use JDK6's javax.ws.rs package
  - http://java.dzone.com/articles/restful-web-services-java

Project Preliminaries

- You will be give Amazon AWS credits by Tue, Jan 14...*provided you send teams by Sun Jan12*
  - Account login & access instructions by email
  - ~$100 per team
- Soft quotas i.e. not physically enforced
- Honour system
- If one overshoots, rest suffer
  - **It will lead to grade reductions, including failing grade!**
  - **Both teammates are responsible!**
- *We will discuss Project 0 on Wed 15 Jan 1-2PM special session in room TBD*