

15-319 / 15-619

Cloud Computing

Recitation 6

October 6th & 8th, 2015

Overview

- **Administrative issues**
Office Hours, Piazza guidelines
- **Last week's reflection**
Project 2.2, OLI Unit 3 module 7, 8 and 9
- **This week's schedule**
 - Project 2.3 - October 11, 2015
 - Quiz 5 - October 9, 2015 (Modules 10, 11, 12)
 - Make 3 person teams for the 15619 Project
- **Demo**

Announcements



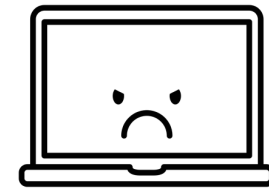
- Monitor AWS expenses regularly
 - Check your bill frequently (use Cost Explorer and filter by tags).
- Terminate your resources when not in use
 - **Stop still costs EBS money (\$0.1/GB/Month)**
 - Amazon EC2 and Amazon Cloudwatch fees for monitoring, ELB
 - Autoscaling group - no additional fees
- Use spot instances

Last Week's Reflection



- Content
 - Unit 3 - Modules 8, 9 and 10:
Virtualizing Resources on the Cloud
 - Quiz 4 completed
- EC2 APIs
 - Amazon CLI, Java, Python
- Load Balancing and AutoScaling
 - Experience ASG (Horizontal Scaling) on AWS
 - Manage cloud resources and deal with failures using programs.

Types of Failures



heroku :: status

APP OPERATIONS TOOLS

Elevated Error Rates

ISSUE: We are continuing to monitor performance issues affecting some customers. We have made changes which should mitigate the problem for most customers and are continuing to monitor the situation.
FEB 08, 2011 – 21:38 UTC – 16 MINUTES AGO

ISSUE: We are continuing to monitor sporadic reports of poor performance.
FEB 08, 2011 – 21:11 UTC – 42 MINUTES AGO

ISSUE: We are continuing to investigate poor performance in the platform.
FEB 08, 2011 – 20:40 UTC – 74 MINUTES AGO

ISSUE: We are continuing to investigate poor performance in the platform.
FEB 08, 2011 – 20:40 UTC – 74 MINUTES AGO

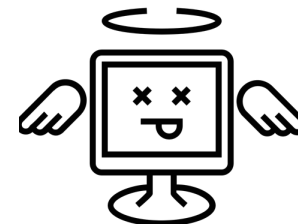
ISSUE: We are investigating elevated error rates on the platform.
FEB 08, 2011 – 20:09 UTC – 105 MINUTES AGO



Transient Failure



Permanent Failure

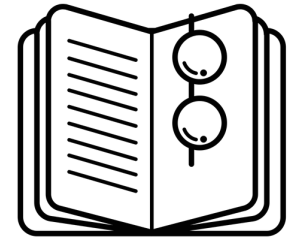


Project 2.2



- Manual Grading: 20 Points are for the code
 - Always make sure that your code is readable
 - Follow style presented in Recitation 2
 - Use the [Google Code Style](#) guidelines

This Week: Content



- UNIT 3: Virtualizing Resources for the Cloud
 - Module 7: Introduction and Motivation
 - Module 8: Virtualization
 - Module 9: Resource Virtualization - CPU
 - Module 10: Resource Virtualization - Memory
 - Module 11: Resource Virtualization – I/O
 - Module 12: Case Study
 - Module 13: Network and Storage Virtualization

Unit 3 : Module 10, 11, 12



- Memory Virtualization
 - Two-level mapping
 - Overcommitment and reclamation
 - [Ballooning](#)

- I/O Virtualization
 - Device Sharing (cross-OS)
 - Privileged Instruction vs Memory-mapped
 - Intercepting I/O requests

- Case Studies and Comparison

Virtualization Black-belt?



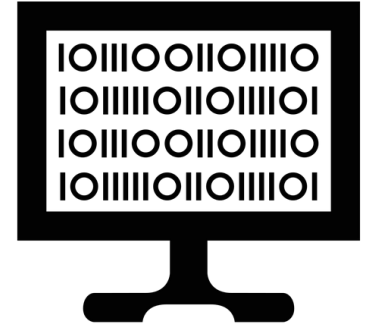
- Goldberg, Robert P. "Survey of virtual machine research." Computer 7.6 (1974)
- Bressoud, Thomas C., and Fred B. Schneider. "Hypervisor-based fault tolerance." ACM Transactions on Computer Systems (TOCS) 14.1 (1996): 80-107.4-45.
- Chen, Peter M., and Brian D. Noble. "When virtual is better than real [operating system relocation to virtual machines]." Hot Topics in Operating Systems, 2001
- Sugerman, Jeremy, Ganesh Venkitachalam, and Beng-Hong Lim. "Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor." USENIX Annual Technical Conference, 2001.
- Barham, Paul, et al. "Xen and the art of virtualization." ACM SIGOPS Operating Systems Review (2003)
- Bellard, Fabrice. "QEMU, a Fast and Portable Dynamic Translator." USENIX Annual Technical Conference, FREENIX Track. 2005.
- Clark, Christopher, et al. "Live migration of virtual machines." Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2. USENIX Association, 2005.
- Rosenblum, Mendel, and Tal Garfinkel. "Virtual machine monitors: Current technology and future trends." Computer 38.5 (2005): 39-47.
- Kivity, Avi, et al. "kvm: the Linux virtual machine monitor." Proceedings of the Linux Symposium. 2007.
- Soltesz, Stephen, et al. "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors." ACM SIGOPS Operating Systems Review. Vol. 41. No. 3. ACM, 2007.
- Bailey, Michelle. "The economics of virtualization: Moving toward an application-based cost model." International Data Corporation (IDC), Whitepaper (2009).
- **Research Areas:** Trust, Security, Patching, Scheduling, Live Migration, Monitoring, Nesting, Networks, Energy Efficiency

Diversion: Containers

- Radically changing software deployment
- Encapsulate application and all dependencies
- Why Containers (not VMs)?
 - Improved utilization
 - Faster provisioning
 - Easier management
 - Microservices
- Why not Containers?
 - Reduced Isolation

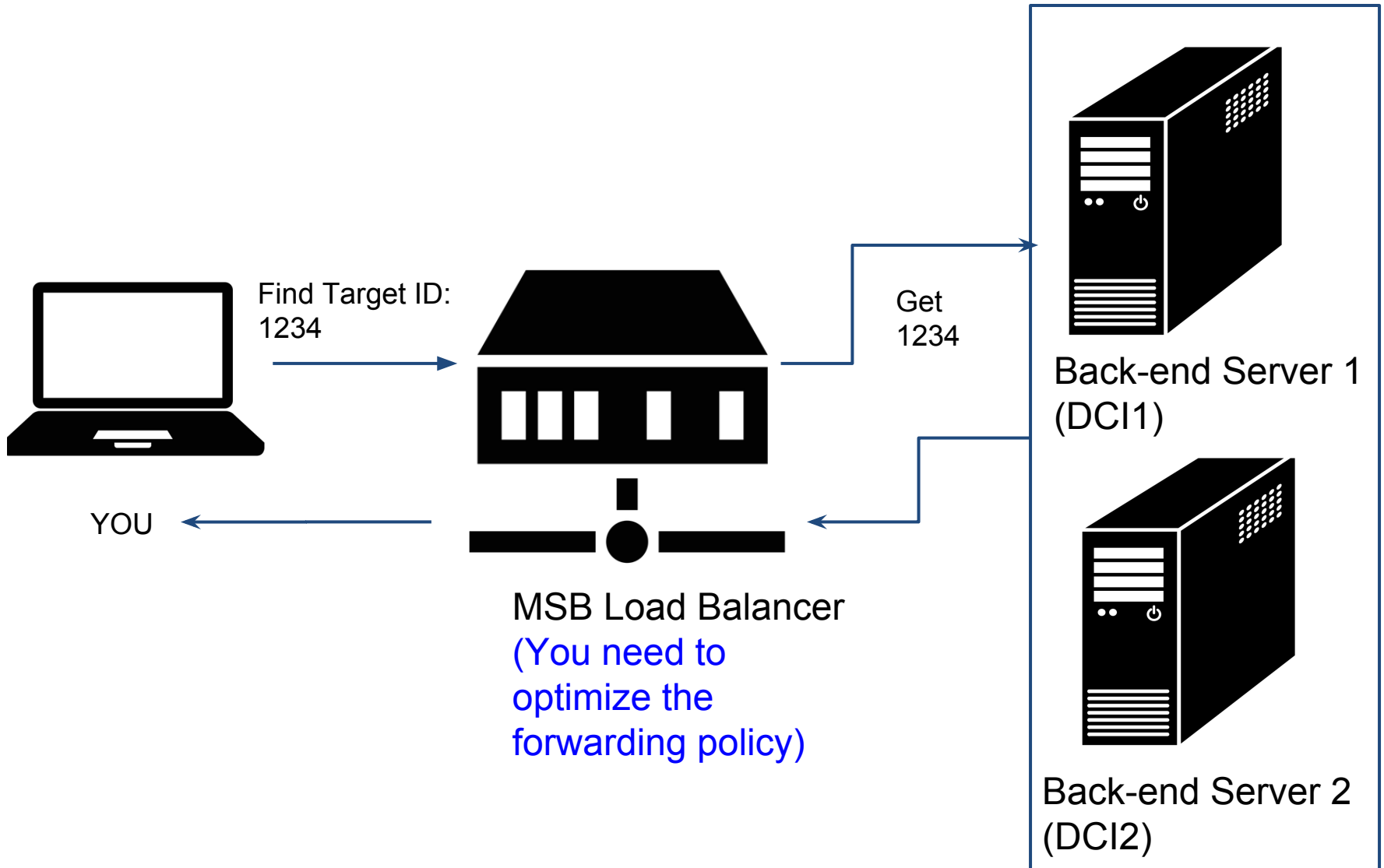


This Week: Project

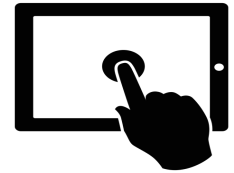


- P2.1: Introduction and APIs
 - MSB Recruitment Exam
- P2.2: Autoscaling and Elastic Load Balancing
 - Junior System Architect at the MSB
- P2.3: Advanced Scaling Concepts: Load Balancer
 - Senior System Architect at the MSB

P2.3 - This Week



Project 2.3: Load Generator UI

A screenshot of a web browser window. The browser's address bar shows a URL with a colorful, pixelated pattern. The main content area of the browser displays a welcome message and a list of steps for using the MSB Load Generator & Test Center. The text is as follows:

Welcome to MSB Load Generator & Test Center, [redacted]!

Step 0. [Enter your submission password](#)

Step 1. [Round Robin Test](#)

Step 2. [Custom Scheduling Algorithm Test](#)

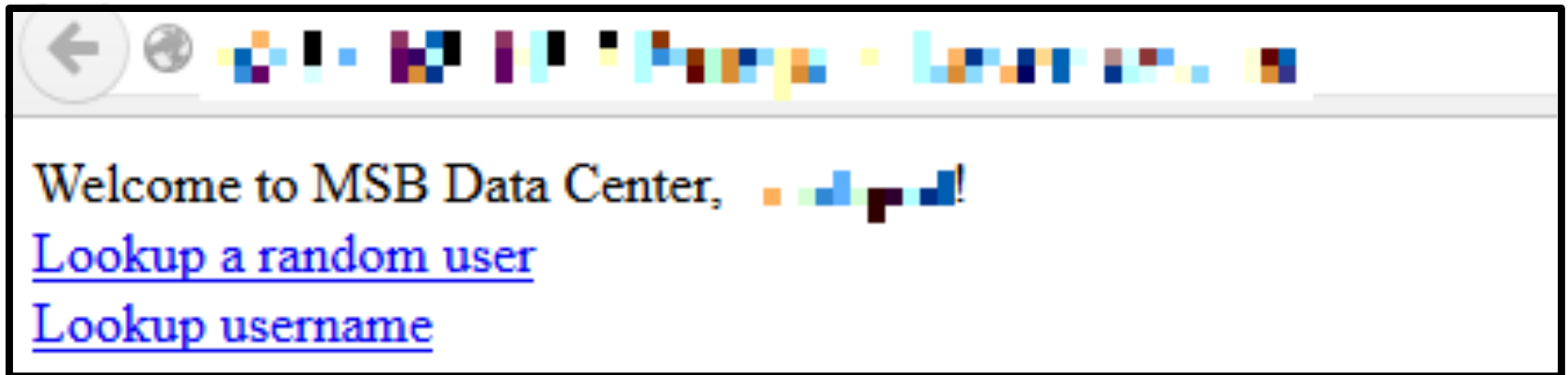
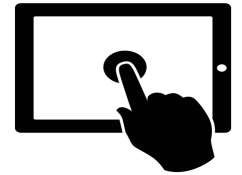
Step 3. [Health Check Test](#)

Step 4. [Senior System Architect Test](#)

Step 5. [Upload Code](#)

[Test logs](#)

Project 2.3: Data Center UI

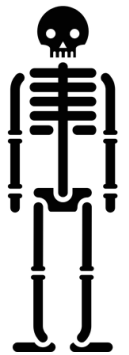


P2.3 - what you have to do



- Write a load balancer that:
 - Uses a simple round-robin algorithm to route traffic
 - Scales-out or scales-in instances based on any detected failures
 - Implements a health-check to detect connected web servers
 - If requests are not heterogeneous-- intelligent routing
 - Combine all the features above and get promoted to be a Senior System Architect at the Mass Surveillance Bureau

Skeleton Code Provided



Skeleton code in Java (`/loadbalancer/`)

- Implements “the plumbing” of a load balancer
 - Setting up of sockets to the client and server
 - API for forwarding requests and receiving responses
- Pending tasks:
 - Implement a simple round-robin request router
 - Implement a health-check that detects and recovers from failure
 - Understand the different types of requests and learn to monitor web servers in real-time
 - Forward requests based on observed load on each web server [in `start()` method]

Project 2.3 Penalties






Project Grading Penalties

The following table outlines the violations of the project rules and their corresponding grade penalties for this project.

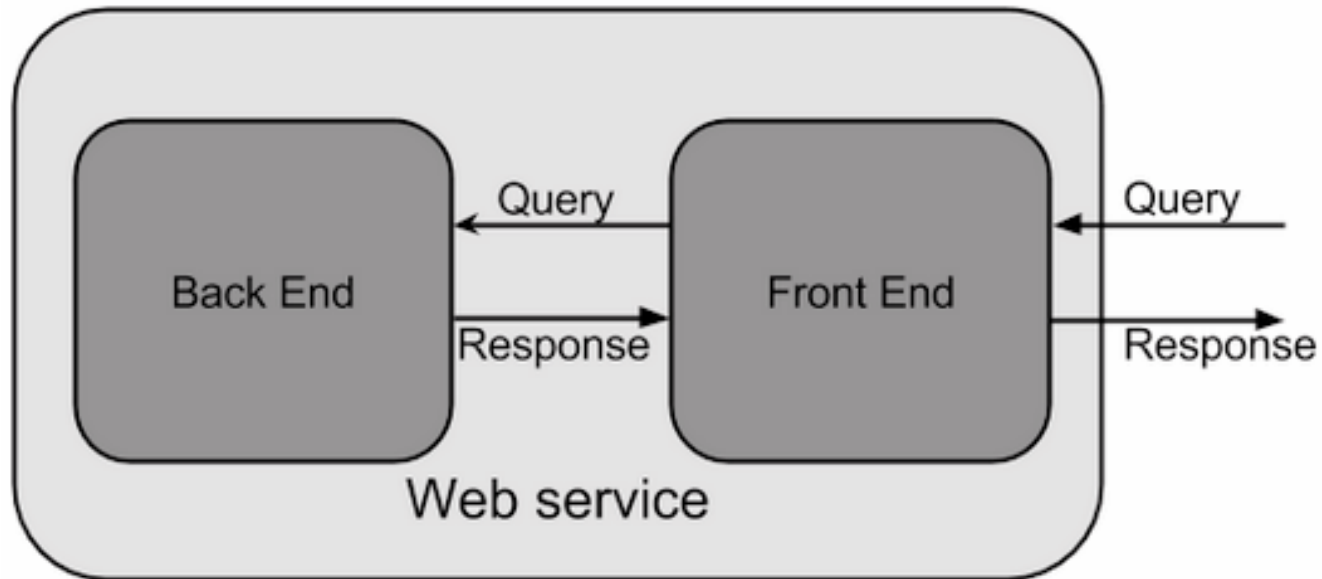
Violation	Penalty of the project grade
Failing to tag your all your instances for this project	-10%
Attempting to hack/tamper the autograder in any way	-100%
Using more than \$10 to complete this project	-10%
Using more than \$20 to complete this project	-100%

Upcoming Deadlines



- Project 2.3: Load Balancer Internals
Due: **10/11/2015 11:59PM Pittsburgh** 
- Quiz 5: Modules 10 to 12:
Due: **Friday 10/09/2015 11:59PM Pittsburgh** 
- 15619Project Team Formation Deadline
Due: **Sat 10/10/2015 11:59PM Pittsburgh** 

15619 Project Architecture

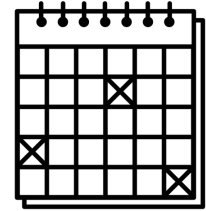


- Writeup and Queries will be released on Wednesday, October 14th, 2015
- We can have more discussions in subsequent recitations
- For now, ensure 3-person teams you decide have experience with web frameworks and database, storage principles and infra setup/hacking

Motivations and End-Goals

- The C10k/C1M Challenge
- Scalable System Design
- Building 1-click clouds
- Resource Allocation
- Distributed and NoSQL DBs [Tradeoff Eval]
- Data Wrangling / Schema design
- Security

15619 Project Time Table

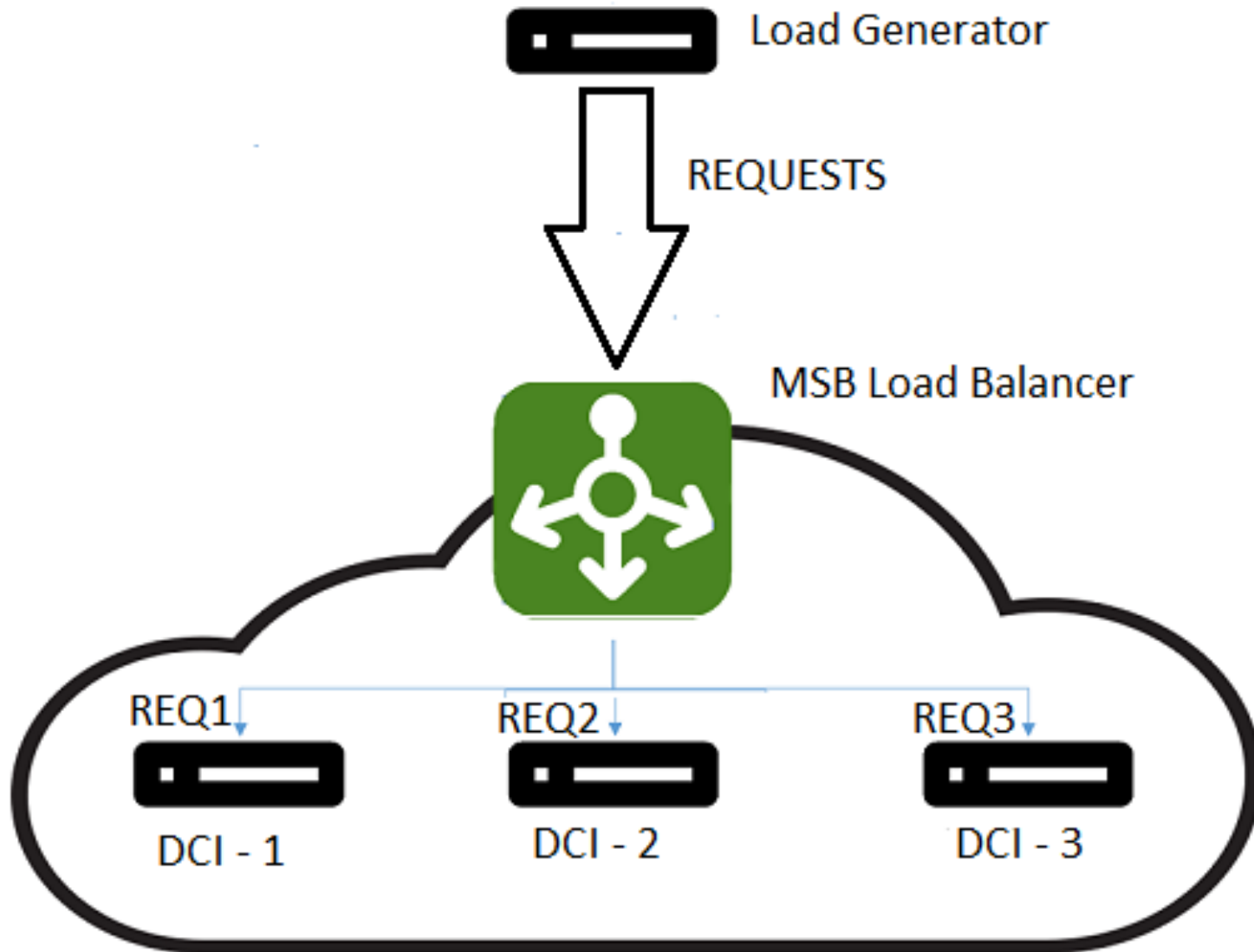


Phase (and query due)	Start	Deadline	Code and Report Due
Phase 1 Part 1 <ul style="list-style-type: none"> Q1 (due), Q2 (not yet due) 	Thursday 10/15/2015 00:00:01 EDT	Wednesday 10/21/2015 23:59:59 EDT	
Phase 1 Part 2 <ul style="list-style-type: none"> Q1, Q2 (due) 	Thursday 10/22/2015 00:00:01 EDT	Wednesday 10/28/2015 23:59:59 EDT	Thursday 10/29/2015 23:59:59 EDT
Phase 2 <ul style="list-style-type: none"> Q1, Q2, Q3, Q4 	Thursday 10/29/2015 00:00:01 EDT	Wednesday 11/11/2015 16:59:59 <u>EST</u>	
Phase 2 Live Test <ul style="list-style-type: none"> Q1, Q2, Q3, Q4 	Wednesday 11/11/2015 18:00:01 <u>EST</u>	Wednesday 11/11/2015 23:59:59 <u>EST</u>	Thursday 11/12/2015 23:59:59 <u>EST</u>
Phase 3 <ul style="list-style-type: none"> Q1, Q2, Q3, Q4, Q5, Q6 	Thursday 11/12/2015 00:00:01 <u>EST</u>	Wednesday 12/2/2015 18:59:59 <u>EST</u>	
Phase 3 Live Test <ul style="list-style-type: none"> Q1, Q2, Q3, Q4, Q5, Q6 	Wednesday 12/2/2015 20:00:01 <u>EST</u>	Wednesday 12/2/2015 23:59:59 <u>EST</u>	Thursday 12/3/2015 23:59:59 <u>EST</u>

Demo

- Load Balancers
 - Motivation
 - P2.3 Load Balancer

P2.3 Load Balancer



Load Balancer - Motivation

- Improved Quality of Service (QOS)
 - Increased throughput
 - Decreased latency
- High Availability (HA)
 - Service Level Agreements
 - Available for close to 100% of the time
 - Load distribution across multiple data centers

Load Balancer - Evaluation

Load Distribution

- Round Robin
 - Homogeneous load performance
 - Heterogeneous load performance
- Something more intelligent?
 - Random? (probably not)
 - Based on request execution time?
 - Based on resource utilization?

P2.3 - Load Balancer

- Health check
 - Measure health of instances
 - Faulty instances will bring down performance.
 - Handle failure of instances efficiently
 - Stop sending requests to failed instance
 - Launch a new instance and add to LB

P2.3 - Load Balancer

- P2.3 Tasks
 - Write code for round-robin scheduling
 - Implement an effective load distribution strategy
 - Implement health check
- All code to be written in the load balancer
- Skeleton code given (Including an API for CPU utilization)

The End