Queen Mary
University of London

ECS781P: Cloud Computing Lab Instructions for Week 6 (Assessed)

Version 2.0

SLA, SLO, Pricing, Basic Service Quality Measurements

Dr. Arman Khouzani

Feb 14, 2017

Instructions

1. This exercise is composed of 3 parts. The solutions need to be submitted through QMplus (link will be emailed to you later today).
2. `Turnitin` service will be used to measure against plagiarism.
3. The submission involves a report that **must be in pdf format** and the codes which must be in their code format.
4. The codes should be saved as separate files than the report. The report does not have to include the code (unless you feel it helps the narrative).
5. The deadline for submission is Wednesday 22 Feb at 12:00 Noon.
6. You are free to use any source as long as you clearly cite it in your report.
7. Collaboration on thinking about the material is encouraged, but you may not copy from other people's works without citation.
8. There is no restriction on the choice of the programming language, but the codes need to run (will be tested).
9. The report can be very short (no need for long explanation). The criteria is clarity and "density" of the material.
10. Any questions can be emailed to me: arman.khouzani@qmul.ac.uk.

11. These exercise has a total of 20 points, but each part has a potential score of 10 points. So there is 10 points leeway.

Question 1: IaaS

1. Create a table: On the first column, pick your 4 choices (arbitrary) from the Amazon's EC2 virtual server instance types (list can be accessed from here or here). Note that for instance `t2.medium` and `t2.large` are two different instance types. For the columns, choose **3 (three!)** of the relevant Quality of Metrics measures introduced in the last lecture. Note that the appropriate choice for "service performance metrics" is the server capacity here, and the server capacity is itself composed of four values: (1- number of cores, 2- clock rate of each core (in GHz), 3- RAM size in GB, 4- Storage size in GB). In other words, these 4 would just constitute ONE column, that is, server capacity. So you will need **2 more columns.** Through AWS documentations or any other resource, fill in the table. **Note that each entry must be accompanied with its appropriate unit.**

2. Provide a short explanation of the following four different costing *On-Demand*, *Reserved Instances*, *Spot Instances* and *Dedicated Hosts*. This is not part of the table. Just write them separately in a listed paragraph like the following:

   - **On-Demand Pricing** description
   - **Reserved Instance Pricing** description
   - **Spot Instance Pricing** description
   - **Dedicated Host Pricing** description

   Add these four columns to the previous table. Specifically, add four columns, each containing the price of that instance in that pricing model with the appropriate unit.

3. Estimate the cost of obtaining the physical version of each of these instances and add as a column. Then a final column to compute how many days of constant usage will equate the cost of physically buying the server.

Question 2: SaaS

1. Write a code (a function, method, etc) that computes the response time to an API request. So the input is the url address of an API and the (can be any language, e.g. Pythong. PHP, Java). You can get hint from the code of the last week (in Python).

2. Use that code to estimate the average, median, maximum and the standard deviation of the response time to your favourite API. A histogram graph (representing the distribution of the response time) can be optionally added to your report as well.

3. Note: the service providers usually have "rate-limiting" mechanisms in place for their API access. For instance, there can be a cap on the maximum number

of API requests per minute. What is the effect of this rate limiting mechanism on your measurements? Modify your code that takes this cap as a parameter and change its behaviour to more accurately compute the API response time under weak loads.

4. Find out what the maximum API rate on your choice of API is and run your measurements again with your modified code.

Question 3: PaaS

This exercise is optional. Using `openshift.eecs.qmul.ac.uk` and the rhc command, write a code that computes the average and maximum instantiating time for your choice of resource. The code can be written in PHP or Shell Script that invokes `rhc`.

Hint: make sure you have set up rhc correctly on your ITL machine: `rhc setup`, and then entering `openshift.eecs.qmul.ac.uk` when prompted, and then your `eecs` username and password. Then run `rhc --help` to see which control commands are available through this command line interface (CLI). You are interested in the commands that can create a resource, make sure it is up and running, and then release that resource, etc.