

PART I: Concepts and Theory, Algorithms [60 points]

Please provide accurate, concise (maximum 10 sentence) answers.

1. Consider equation (12.1, pg 643) from the textbook for the response time of an app. [15 points]

a. What is the typical response time of a web application (research using the web). What is an acceptable range for it, and what problems occur if it exceeds that range.

What is the typical response time of a web application?

For a web application, response time is the time interval between the moment when the user submits a request to the web application and the moment when the user receives a response.

As the equation from the textbook shows:

$$R = \frac{D}{B} + T_{RTT} + T_S + T_C$$

D: the amount of data transfer required to serve the user request

B: the minimum bandwidth across all links in the network from the user to the application deployment

T_{RTT} : the round trip time for user-application interactions that are needed to generate the response

T_S : the total processing time required by all the tiers of the application deployment

T_C : the total processing time required by the user's device.

What is an acceptable range for it, and what problems occur if it exceeds that range?

A one-second response time is generally the maximum acceptable limit, as users still likely won't notice a delay. Anything more than one second is problematic, and with a delay of around five or six seconds a user will typically leave the website or application entirely.

b. Consider 3 different types of workloads from this overview:

CPU Intensive, Memory Intensive and Storage Intensive. Provide real world examples of each of these. Comment on how the different Parameters on the right-hand side of the equation (12.1) will be different for each of these workload type and why. Please explain.

Provide real world examples:

CPU Intensive: application of tasks of sorting, search, graph traversal, matrix multiply, Video editing and processing · Software compiling.

Memory Intensive: web browsers, 3D graphics, IDEA/Eclipse/PyCharm

Storage Intensive: database applications (such as MySQL, IBM DB2, AWS DBS)

For the parameters on the right-hand of the equation:

D: the amount of data transfer required to serve the user request

B: the minimum bandwidth across all links in the network from the user to the application deployment

T_{RTT} : the round trip time for user-application interactions that are needed to generate the response

T_S : the total processing time required by all the tiers of the application deployment

T_C : the total processing time required by the user's device.

CPU Intensive:

the amount of data transfer required to serve the user request would be higher;

the total processing time required by all the tiers of the application deployment would be lower;

Memory Intensive

the round trip time for user-application interactions that are needed to generate the response would be lower;

Storage Intensive

D/B could be lower as storage intensive application could handle large amount of data transfer required to serve the user request quicker.

2. Write a 1-2-page Design Note on how you would design a Performance Testing project for your eCommerce SaaS on AWS Cloud. Please refer to the paper Mukherjee et al. Performance Testing Web Applications on the Cloud, and provide similar approach, with diagrams. [15 points]

TEST PROCEDURE

For my eCommerce SaaS on AWS Cloud, I would use three types of AmazonEC2 instances: small, medium and large. The characteristics of these instances are as follows:

Instance Type	CPU Units	Cores	Memory [GB]	Disk [GB]	Cost [\$ /h]
Small	1	1	1.7	160	0.007
Medium	2	1	3.7	320	0.013
Large	4	2	7.5	850	0.026

All these instances are located in the same region (us-west1) and their IP addresses belong to the same network subnet.

Ubuntu server version 12.04 (kernel version 3.2.0-40-virtual) is installed as the Operating System in all three instances.

Apache 2 (version 2.2) is installed as the Web server.

PHP with FastCGI (version 5.3.6) as the application tier.

MySQL (version 5.1.49-1) as the database server in all instances.

Collectd is used in each instance to monitor the CPU utilization.

The httpperf Web workload generator tool [7] is used to generate synthetic requests to the Web Server.

Server response time is the only focused metrics in my eCommerce SaaS.

I will conduct a subset of experiments over a week to look for performance variability in the EC2 platform. In the performance Testing project, httpperf is run at the beginning of each hour for 24 hours in a day over the course of a week. Each httpperf test to an EC2 instance runs for 100 seconds. Performance data from the instances was captured only once every hour of the day for a week in order to minimize the cost of running the EC2 instances continuously.

Steps:

First step: run experiments to rule out bottlenecks in the test setup that can prevent us from inferring the true impact of the EC2 platform on Web server response time. (Specifically, I will conduct experiments for determining whether the workload generator should be located inside the cloud or in a dedicated physical host in a remote location) Based on these results, it was clear that the performance impact of the EC2 platform is better evaluated when the workload generator resides inside the cloud.

Second step: conduct experiments to determine the right type of EC2 instance to serve as the workload generator. The experimental results show how choosing a wrong instance as the workload generator can badly affect our estimates of server performance.

Third step: I will continuously monitor the network bandwidth for a week between the workload generator and the Web server instances under test using the Iperf tool.

Forth step: analyze the results to see if there is a wide variation in available bandwidth and whether the available bandwidth is sufficient to sustain our test workloads.

Finally step: run experiments to analyze the performance of the three types of EC2 Web server instances

VALIDATING THE TEST PROCEDURE

A. Using a Remote Workload Generator

Internet round trip latencies dominate the response times measured by httpperf.

CPS	Small-R	Small-C	Med-R	Med-C	Large-R	Large-C
100	26.9	25.7	26.3	25.3	26.9	25.2
200	26.9	25.5	26.5	25.2	27.1	25.3
300	28.9	27.2	27.0	25.2	29.2	27.4
600			29.5	25.2	29.6	28.4
1000					32.1	33.4
1200					34.5	33.8

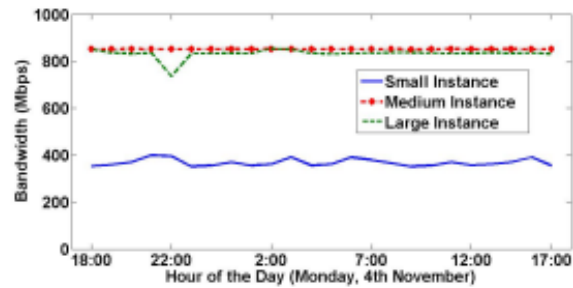
B. Using In-Cloud Workload Generator

The performance of each EC2 instance is measured in terms of response time (in milliseconds).

Workload Generator	CPS	Small Instance	Medium Instance	Large Instance
small	100	14.7	11.5	5.9
	200	22.0	11.8	6.6
	300	100.7	18.0	7.1
	600		89.4	15.6
	1000			30.1
	1200			36.2
medium	100	12.1	7.6	4.5
	200	19.6	10.3	4.7
	300	93.4	15.2	5.6
	600		80.1	10.1
	1000			19.2
	1200			28.4
One large	100	10.9	5.2	3.6
	200	17.2	8.2	3.8
	300	87.6	12.3	3.9
	600		73.3	5.0
	1000			7.3
	1200			12.1
Two large	500			7.4
	600			12.3

C. Measuring Network bandwidth for Tests

The available bandwidth between the workload generator and the small instance shows some variations for different hours of the day.



3.

a. Make 2 or 3 basic tables (only few example rows) for your eCommerce App (SaaS) following these schemata. What are the advantages of using postgresQL for this data?

user
user_id
username
password
name
phoneNum
createdTime
address

user_id	username	password	name	phoneNum	createdTime	address
0	Hello	123	Jason	6693084725	2021-02-17	A St.
1	World	258	Edward	6673089753	2021-05-06	32 way
2	Cloud	1314	Sarah	6587222587	2020-11-23	B St.

cart_item
user_id
session_id
product_id
price

user_id	session_id	product_id	price
0	258	69	29.99
1	467	945	3.65
2	32	364	12.8

What are the advantages of using postgresQL for this data?

Postgres is an object-relational database, while MySQL is a purely relational database. This means that Postgres includes features like table inheritance and function overloading, which can be important to certain applications. Postgres also adheres more closely to SQL standards. Postgres handles concurrency better than MySQL for multiple reasons: Postgres implements Multiversion Concurrency Control (MVCC) without read locks Postgres supports parallel query plans that can use multiple CPUs/cores Postgres can create indexes in a non-blocking way (through the CREATE INDEX CONCURRENTLY syntax), and it can create partial indexes (for example, if you have a model with soft deletes, you can create an index that ignores records marked as deleted) Postgres is known for protecting data integrity at the transaction level. This makes it less vulnerable to data corruption.

b. How do you map (translate) this schema to a Graph Db schema? Answer this question using the reference resources.

STEP ONE:

The first step is to identify the data entity types. In my schema there are two types of data entities: user and cart items

STEP TWO:

In the second step, I need to find the semantic relations between those entities.

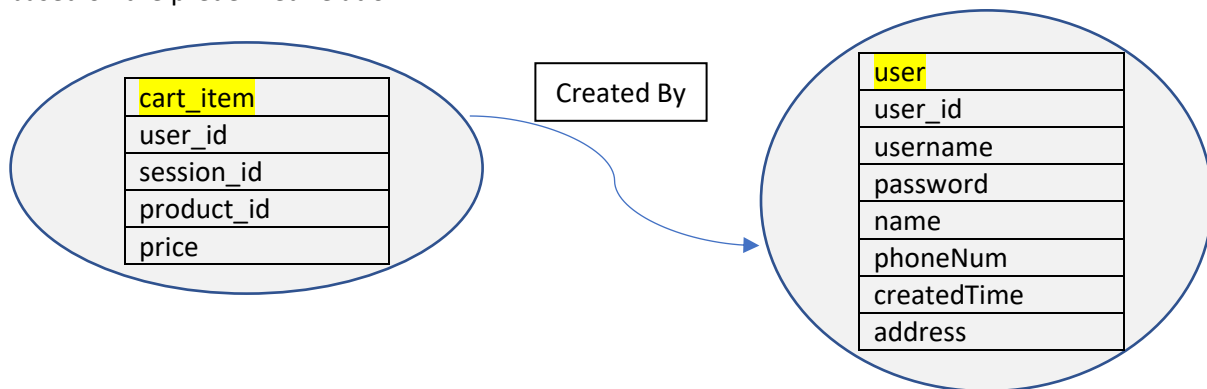
We find five relations between them which are:

Relation: Cart items – CreatedBy -> User

Logical Link: User_id

STEP THREE:

When data entities and relations are correctly identified, we can create nodes for each data entity and edges for each relation we just discovered. Now we've created a graph from the relational data model based on the predefined relation.

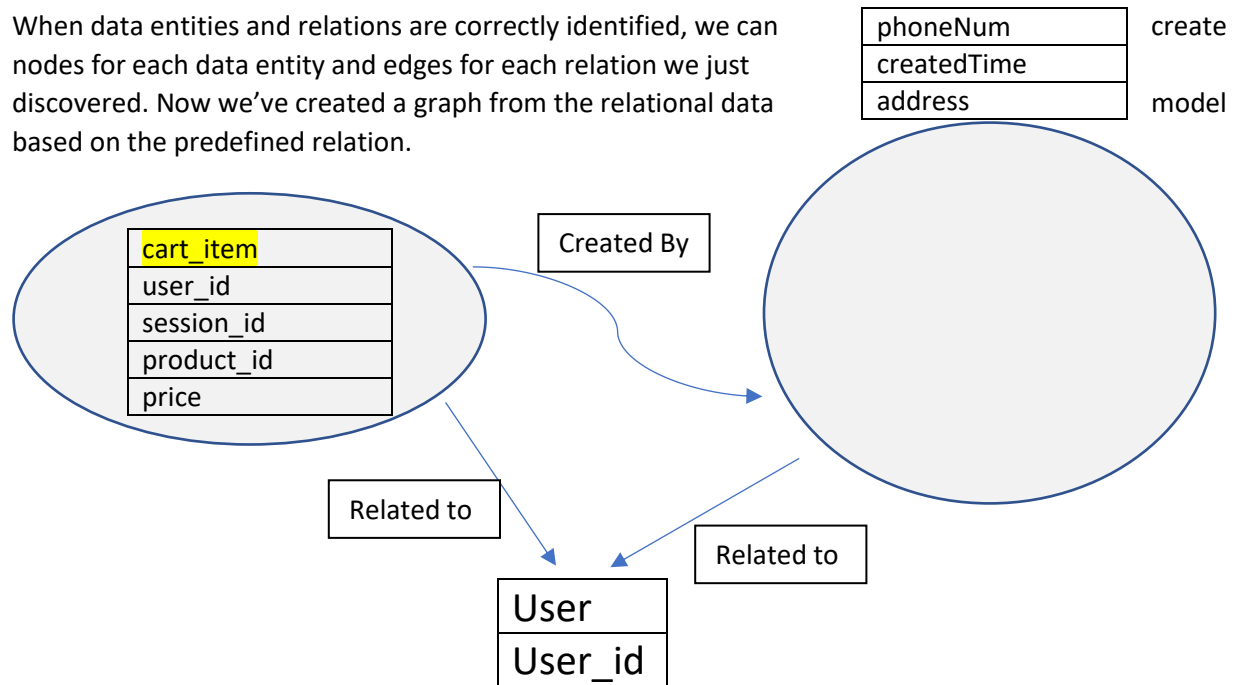


STEP FOUR:

Promoting those common attributes to nodes with proper relations to the entities could add value to my graph.

user
user_id
username
password
name

When data entities and relations are correctly identified, we can nodes for each data entity and edges for each relation we just discovered. Now we've created a graph from the relational data based on the predefined relation.

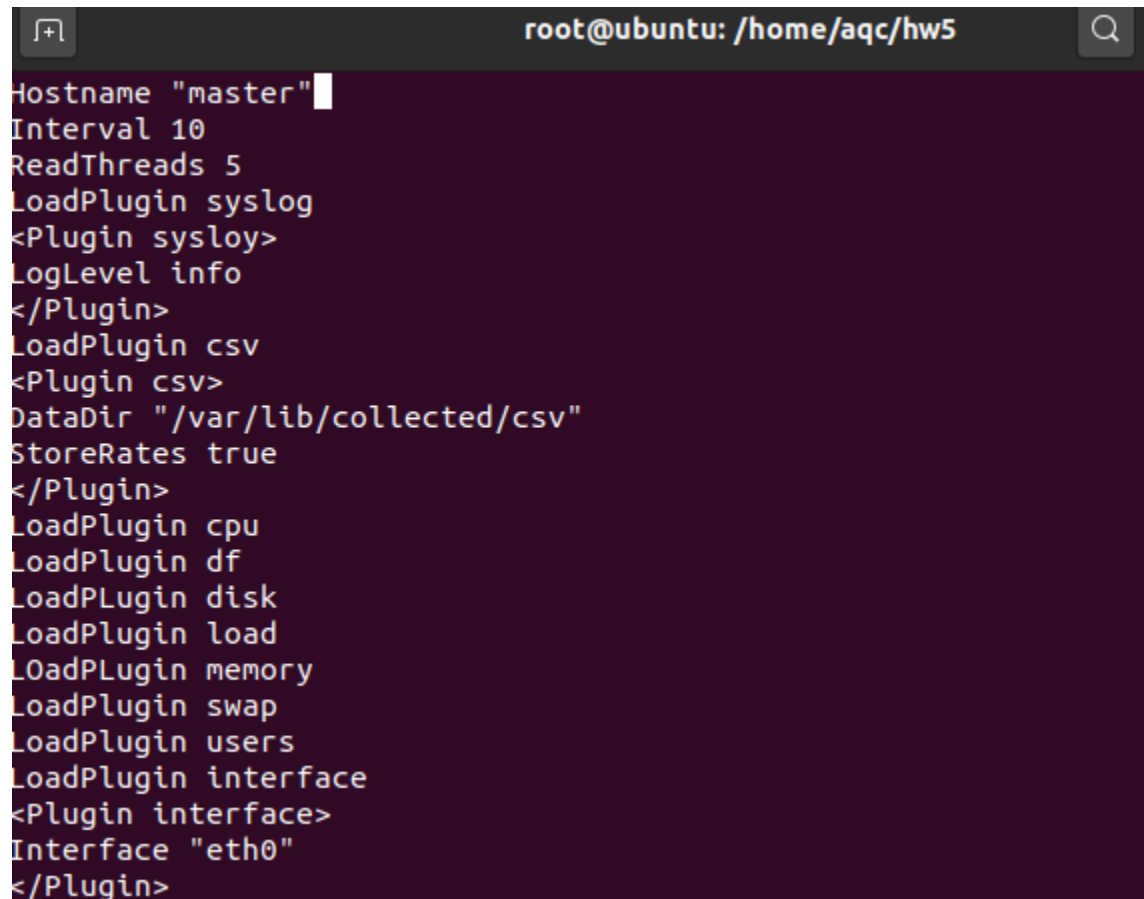


PART II: LAB [30 points]

Collectd

```
root@ubuntu:/home/aqc/hw5# apt-get install collectd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

```
root@ubuntu:/home/aqc/hw5# vim /etc/collectd.conf
```



```
root@ubuntu: /home/aqc/hw5
Hostname "master"
Interval 10
ReadThreads 5
LoadPlugin syslog
<Plugin syslog>
LogLevel info
</Plugin>
LoadPlugin csv
<Plugin csv>
DataDir "/var/lib/collectd/csv"
StoreRates true
</Plugin>
LoadPlugin cpu
LoadPlugin df
LoadPlugin disk
LoadPlugin load
LoadPlugin memory
LoadPlugin swap
LoadPlugin users
LoadPlugin interface
<Plugin interface>
Interface "eth0"
</Plugin>
```

```
root@ubuntu:/home/aqc/hw5# systemctl enable collectd
Synchronizing state of collectd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable collectd
```

```
root@ubuntu:/home/aqc/hw5# systemctl start collectd
```

```

root@ubuntu:/home/aqc/hw5# ls /etc/init.d/
acpid          grub-common    procps
alsa-utils     hwclock.sh     pulseaudio-enable-autospawn
anacron        irqbalance     rsync
apparmor       kerneloops     rsyslog
appport        keyboard-setup.sh saned
avahi-daemon   kmod           speech-dispatcher
bluetooth      lvm2           spice-vdagent
collectd       lvm2-lvmpolld  udev
console-setup.sh network-manager ufw
cron           open-vm-tools  unattended-upgrades
cups           openvpn        uidd
cups-browsed   plymouth       whoopsie
dbus           plymouth-log   x11-common
gdm3           pppd-dns

```

```

root@ubuntu:/home/aqc/hw5# /etc/init.d/collectd restart
Restarting collectd (via systemctl): collectd.service.
root@ubuntu:/home/aqc/hw5#

```

```

root@ubuntu:/var/lib/collectd/csv/master# ls
cpu-0      df-snap-core18-2246      disk-loop1    disk-sda
cpu-1      df-snap-gnome-3-34-1804-24  disk-loop10   disk-sda1
df-boot-efi df-snap-gnome-3-34-1804-72  disk-loop11   disk-sda2
df-dev      df-snap-gtk-common-themes-1506 disk-loop2    disk-sda5
df-dev-shm  df-snap-gtk-common-themes-1519 disk-loop3    disk-sr0
df-root     df-snap-snapd-13640        disk-loop4    load
df-run      df-snap-snapd-7264         disk-loop5    memory
df-run-lock df-snap-snap-store-433     disk-loop6    swap
df-run-user-1000 df-snap-snap-store-547    disk-loop7    users
df-snap-bare-5 df-sys-fs-cgroup          disk-loop8
df-snap-core18-1705 disk-loop0                disk-loop9

```

Httpf

```

(base) aqc@ubuntu:~/hw5$ conda create -n hw5 python=2.7

```

```

(base) aqc@ubuntu:~/hw5$ conda activate hw5
(hw5) aqc@ubuntu:~/hw5$ pip

```

```
(hw5) aqc@ubuntu:~/hw5$ pip install Httpperfpy
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.7. More details about Python 2 support in pip, can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support
Collecting Httpperfpy
  Downloading https://files.pythonhosted.org/packages/0c/fd/282672ee3bb4be40e89076be22721e056db68ff155a177a4c79afbdf1/httpperfpy-0.0.3.tar.gz
Building wheels for collected packages: Httpperfpy
  Building wheel for Httpperfpy (setup.py) ... done
  Created wheel for Httpperfpy: filename=httpperfpy-0.0.3-cp27-none-any.whl size=5399 sha256=5c1a8678f346b6ca80bab2a7cb457566b59d22d28b324ea5bb71ab7fe29d3474
  Stored in directory: /home/aqc/.cache/pip/wheels/42/22/41/4aab170dd04df1843be4db56e2580c3a9cb06c28d0e6915b0b
Successfully built Httpperfpy
Installing collected packages: Httpperfpy
Successfully installed Httpperfpy-0.0.3
```

```
(hw5) aqc@ubuntu:~/hw5$ vim httpperf.py
```

```
from httpperfpy import Httpperf

for i in range(10,60,10):
    print 'Rate=' + str(i)
    perf = Httpperf(server = "www.mywebsite.com",
                    port = 80, rate = i, num_calls = 50, num_conns = 400,
                    timeout = 60, send_buffer = 4096, recv_buffer = 16384)

    perf.parser = True

    results = perf.run()

    print 'Request rate per sec, ...'
    print results["reply_rate_max"]
```

Final result

```
Rate=10
Request rate per sec, Connection rate per sec, Avg reply rate, Max reply rate, Response Time
20.0, 10.0, 10.0, 10.0, 89.3
Rate=20
Request rate per sec, Connection rate per sec, Avg reply rate, Max reply rate, Response Time
39.8, 19.9, 19.9, 20.0, 87.8
Rate=30
Request rate per sec, Connection rate per sec, Avg reply rate, Max reply rate, Response Time
59.4, 29.7, 29.6, 30.2, 86.2
Rate=40
Request rate per sec, Connection rate per sec, Avg reply rate, Max reply rate, Response Time
78.9, 39.4, 39.4, 40.0, 90.5
```

