CS5630 Midterm Exam

Name:

HawkID/university ID:

- This is a closed book, closed notes, closed electronics, but open minds exam
- You have five mandatory questions (Q1 — Q5), all of which carry 20 points
- There is an extra credit question (Q6). Points earned in Q6 can make up for points lost in other questions, but cannot take your score beyond 100
- Keep your answers brief and to the point. While the space provided here should be sufficient to write your answers, we will provide extra sheets if needed.
- You are welcome to clarify any doubts or concerns with the instructor, but cannot engage in discussions with your fellow students
- We expect you to exhibit highest levels of academic integrity and honesty

Q1. Cloud services **20 points**

We discussed four different levels of abstraction in which compute could be obtained as a service on the cloud.

(a) Name any three of them. For each of these service levels, explain what the cloud provider offers and how the cloud users benefit from them. (12 points)

- **Infrastructure-as-a-Service**: cloud providers offer low-level infrastructure for compute (VMs), storage (block devices), and networking (virtual interfaces); cloud users use them akin to how they use actual hardware in the real-world.
- **Platform-as-a-Service**: cloud providers offer frameworks (e.g., map-reduce) and system software (e.g., database systems) as managed services; cloud users get the advantage that they don't have to administer these systems as well as not have to worry about managing the hardware on which these run.
- **Software-as-a-Service**: cloud providers offer advanced software such as vision, media processing, language processing, and other AI/ML software, etc as managed services; cloud customers simply bring their data to work with these software without having to worry about either the software or hardware on the cloud.
- **Functions-as-a-Service**: cloud providers offer a sandboxed compute environment for running user-specified functions; cloud users get the ability to write custom code that runs in the cloud without having to manage the VMs.

(b) What level of service would you recommend if the workload consists of processing data from IoT sources, exhibits sporadic traffic, needs well-defined but minimal processing per request, and does not rely on any legacy software? Explain your choice. (8 points)

Ideally, **Functions-as-a-Service** since data processing is not continuous (i.e., IaaS VMs will be wasted), and since processing is minimal and doesn't need legacy software (i.e., there are no benefits in using advanced services from PaaS or SaaS).

Q2. Cloud and datacenter concepts **20 points**

Describe the following concepts in 1-2 sentences. Then, using a real-world cloud application, explain how each of these benefit/hinder/influence its design and operation.

(a) Warehouse-scale computing

A paradigm of computing where applications require infrastructure at a massive scale (of the order of entire datacenters) to meet their performance goals. WSC abstraction enables applications such as GMail, Facebook etc to scale to billions of users, and to store and operate on petabytes of data, while maintaining low latency interactivity.

(b) Tail latency

Tail latency refers to the latency of the slowest request i.e., the tail of the latency distribution. An application that depends on a large number of sub-tasks to finish before proceeding to the next phase (for example, map-reduce applications have a barrier after map) could suffer from tail latency of the worst performing sub-task. One way to make such applications tail-tolerant is to replicate sub-tasks.

(c) Graceful degradation

Graceful degradation is the ability of a computing system to continue to operate with limited functionality even when a large portions of its underlying infrastructure has failed or rendered inoperative. Users interacting with applications that are designed to exhibit graceful degradation would not see cascaded/catastrophic failures. For e.g., Twitter loading only a partial timeline, or GMail giving the ability to search an existing mailbox but not allowing sending new mails.

(d) Jevon's paradox (hint: we learnt about it in David Irwin guest lecture)

Jevon's paradox is an observation that increasing the resource efficiency often increases, rather than decreases, that resource's consumption. This has broad applications in cloud economics, where making computing efficient (on the cloud) results in more compute being consumed; or in energy systems, where increasing energy efficiency lowers the cost of energy, which in turn can increase the energy demand.

Q3. Cloud tradeoffs **20 points**

We studied several techniques for making the cloud and its applications fault-tolerant. Name any four such techniques, and describe how they work in 1-2 sentences. For each of these techniques, explain the resulting availability vs. performance tradeoff.

(I) **Replication**: store multiple identical copies of data across different servers. Replication increases throughput and availability, but makes application design more complex while also reducing resource utilization.

(II) **Partitioning/Sharding**: split the data into smaller fragments and distribute them across a large number of servers. Sharding increases throughput and availability but makes application design more complex.

(III) **Load balancing**: spread the incoming request for data access/processing across a number of servers (this assumes the underlying data to be sharded/replicated). Load balancing lowers the latency but increases application complexity.

(IV) **Eventual consistency**: design applications to tolerate inconsistent states for limited periods, provided that the system eventually returns to a stable consistent state. Eventual consistency improves the performance but needs the applications to be robust to transient states.

(V) **Integrity/Health checks**: continuous monitoring of server status and data integrity. Health checks incur additional performance but timely detection of failures help make the applications degrade gracefully (as opposed to experiencing catastrophic failures).

(VI) **Centralized control**: design systems to have centralized control but distributed processing of workload (for example, MapReduce master-worker model). Centralized control makes application design easier but exposes them to single points of failure.

Q4. Cloud research **20 points**

This question covers key research papers discussed/assigned in the class.

(a) What is the "collection view" of datacenter disks? Name any three key metrics of the collection view of disks (out of the five identified in the original paper). (10 points)

The collection view advocates that disks of datacenters are always a part of a large collection of disks, and therefore, any optimization should focus on the whole collection without regard to the individual disks.

Five key metrics discussed in the Google FAST 2016 paper are: IOPS (higher is better), disk capacity (higher is better), tail latency (lower is better), security guarantees (stronger is better), and total cost of ownership (lower is better).

(b) What is energy virtualization? Does it help cloud applications to achieve carbon-efficiency, or energy-efficiency, or cost-efficiency, or all of these? Explain your answer. (10 points)

Energy virtualization, akin to hardware virtualization, is the process of exposing visibility into and allowing control of the underlying energy infrastructure to software applications. The goal of energy virtualization is to allow applications to manage their energy sources and to optimize its consumption, explicitly but easily.

Energy virtualization can help applications achieve carbon-efficiency and cost-efficiency. By designing suitable APIs, the carbon footprint and the cost per joule of each virtualized energy source can be exposed to the applications, thereby allowing it to optimize for both carbon- and cost-efficiency. On the other hand, energy-efficiency is an application specific metric that is independent of the underlying energy sources. Thus, energy virtualization is neither necessary nor useful in making an application energy-efficient.

Q5. Cloud programming **20 points**

We learnt how to access AWS resources programmatically via the Python SDK called boto3. The following two questions are about using boto3 to access and control AWS S3.

(a) An imaginary student of CS5630 class has written the following routine. Explain what the student is intending to do. (8 points)

```
import boto3
s3 = boto3.resource('s3')
course_bucket = s3.Bucket('CS5630')

for my_ticket_to_A_plus in course_bucket.objects.filter(Prefix='solutions'):
    print(my_ticket_to_A_plus)
```

The student is iterating through the S3 bucket called CS5630, looking for all the objects beginning with the prefix "solutions".

(b) The unsuspecting professor has written the following code to store the course material on S3. This code can be changed in several ways to invalidate the imaginary student's routine. Suggest TWO such options (please keep your answer technical). (12 points)

```
import boto3
s3 = boto3.resource('s3')
s3.create_bucket(Bucket='CS5630', CreateBucketConfiguration={'Location':'us-east-1'})

for filename in os.listdir('/usr/professor/cs5630/'):
    with open(os.path.join('/usr/professor/cs5630/', filename), 'rb') as filedata:
        s3.Bucket('CS5630').put_object(Key=filename, Body=filedata)
        s3.Object('CS5630', filename).put(ACL='public-read')
```

The professor can: (1) **skip uploading** all the filenames that begin with "solutions" by adding an if condition right after the for loop, or (2) **modify the ACL** on the last line to be set to ACL='private' for filenames beginning with "solutions" or (3) **create a separate bucket** in S3 for storing files that contain solutions, or (4) **encrypt** the solution files before uploading them to the bucket. Any other reasonable suggestion will also get full points.

Q6. Bonus question **10 points**

In the context of datacenter hardware infrastructure, we learnt that compute and storage scale well horizontally (i.e., if you need extra aggregate capacity, simply add more boxes). However, this does not apply to networking (i.e., simply adding extra bandwidth to a server does not help). Explain why? How do datacenter operators solve this problem?

Networking has no straightforward horizontal scaling solution. Adding extra bandwidth to a leaf server simply increases its bandwidth to its immediately connected router. The assumption in a networked system is that every server needs to talk to every other server; thus, the **bisection bandwidth** of the network (i.e., the bandwidth across the narrowest line that divides the servers into two equal groups) needs to be scaled up.

Next, increasing the bisection bandwidth requires replacing the existing switches/routers with larger ones that can support the increased capacity. This approach has its limitations since at some capacity, we will hit the upper limit of that switch/router. To solve this problem, datacenter operators arrange smaller switches in a cascaded fashion to create larger logical switches. Examples of this are fat tree network and Clos network.