Exam Example - Solutions

Index

Part A

- 1. Cloud Computing Paradigm
  - (a) Essential Characteristics
  - (b) Public vs. Private/Community Clouds
  - (c) Example of IaaS, PaaS, SaaS
- 2. Data Storage
  - (a) Object Stores vs. File Systems
  - (b) Data Warehouses vs. Database Systems
- 3. Cloud Computation
  - (a) Google AppEngine vs. Google Compute Engine
  - (b) Sole Tenant vs. Multi-Tenant Nodes
- 4. PySpark Code Fragment
  - (a) Data Processing Explanation
  - (b) Spark Execution Stages

Part B

- 1. Handling Big Data in the Data Sources Layer
- 2. Limitations of Current Big Data Systems
- 3. Explanation of Python Code
- 4. GPUs advantages and disadvnatages
- 5. Explanation of Python Code
- 6. Representing a Binary Classification Problem
- 7. Multi-threading in Loop Execution
- 8. Differences Between Script Codes

Part A

1. In what concerns the cloud computing paradigm:

(a) Identify 3 essencial characteristics associated to the cloud computing paradigm.

On-demand self-service: A consumer can provision computing resources without human intervention.

Broad network access: The cloud services are available over the network through standard means (e.g. Internet connection, thin clientes such as smartphones or PCs).

Resource pooling: Resources are pooled by the cloud provider to serve multiple consumers using a multi-tenant model.

Rapid elasticity: The amount of computing resources can be adjusted/scaled dynamically at the consumer's request or automatically in some configurable manner by the consumer.

Measured service: Resource usage can be tracked and measured precisely by both the cloud provider and consumer, for instance to allow for a clear billing analysis.

(b) What is the distinction between public clouds and private/community clouds? State an advantage and a disadvantage of using a public cloud.

Public cloud:

- Open for use to the general public by a cloud provider. The cloud provider may be a business operator (e.g. Google) or a public institution.
- **Advantage:** Cost-effective.
- **Disadvantage:** Security concerns.

Private / community clouds:

- Used exclusively by of a single organization (private) or a group of organizations (community).

Hybrid cloud results from the combination of two or more clouds that exist separately but are interoperable in some way.

(c) Provide an example of cloud service for each of the IaaS, PaaS, e SaaS service models. Explain your choices.

Infrastructure-as-a-Service (IaaS)

- Customers get actual infrastructure in the form of computing resources (e.g., for processing, storage, or networking purposes).
  The service provider hosts, maintains, and updates the backend infrastructure, such as compute, storage, networking, and virtualization.
  Customers manage everything else including the operating system, middleware, data, and applications.
- **Example:** Google Compute Engine
  - Main parameters: Name, Region and Zone (defines the hosting data center), Machine Type (base CPU + memory configuration), and Boot disk (initial OS image + disk size).

Platform-as-a-service (PaaS)

- Customers get a platform that supports the development and/or deployment of cloud applications (e.g., database engines, data workflows, programming environments).
  As with IaaS models, in PaaS models, the service provider delivers and manages the backend infrastructure. Additionally, PaaS models provide all the software features and tools needed for application development.
  The customer manages your applications and data but need not be concerned with managing or maintaining the software development platform.
- **Examples:** database Engines, programming "notebooks", cloud "functions"
  - Services are configured for the development/deployment of cloud applications. Little infrastructure configuration is required: the infrastructure is deployed automatically according to high-level parameterisation.

Software-as-a-Service (SaaS)

- Customers get access to applications running in the cloud (e.g., Gmail, Google docs). The cloud is transparent to the user.
  The service provides manages everything, including the application and data.
  The customer simply connects to the service through the Internet.
- **Examples:** Google Maps, Skype, Dropbox, Linkdin
  - Software deployed on the cloud that is available to general users.

2. Regarding data storage, state 2 fundamental differences between the use of:

(a) "Object stores" (e.g. buckets in Google Cloud) vs. file systems.

File systems

- Tied to a particular hierarchical organisation and storage device
- Files can be modified or accessed arbitrarily
- Size limit determined by underlying storage facility
- Accessed by programs running in a host that mounts the file system
- Can be mounted efficiently over the network by a host

Buckets

- Data storage not bound logically to any storage facility
- Whole-object operations
- Has no size limit even if contained objects do
- Accessible over the Internet
- Can be mounted over the network by a host (e.g. using gcsfuse) but with performance limitations

(b) Data warehouses like BigQuery vs. database systems.

Data warehouses systems used for storing and querying large datasets. They are used by data analytics tools whose purpose is to run intensive queries over the data in scalable manner.

In contrast to DBMS:

- stored data is immutable (write-once or append-only)
- system must scale for extremely large datasets (up to penta/hexa-byte scale)
- queries typically involve collection of records (aggregate queries)

3. Regarding cloud computation:

(a) Suppose you wish to implement a cloud application and that you have an option between using 1) Google AppEngine and 2) dedicated virtual machines using Google Compute Engine. Explain the difference from the perspective of resource management and billing.

App Engine: provides a more managed approach, ideal for rapid application development and deployment with automatic resource scaling and pay-per-use billing for efficient cost management.

Compute Engine: offers greater control and flexibility, but requires more manual configuration and potentially higher costs if resources are not optimized.

(b) What is the difference between sole-tenant and multi-tenant nodes in a cloud data center? What kind of benefits may a a sole-tenant node bring?

Multi-Tenant Nodes: These are physical servers shared by multiple customers running virtual machines. Each customer gets a virtualized slice of the resources.

Sole tenancy: a client may reserve a server machine that is exclusively dedicated to running the client's VMs. Sole-tenant modes can be managed entirely by the client providing a more isolated environment, more stable performance, and efficient resource sharing between VMs.

4. Consider the following Pyspark code fragment, a variant of the classic "word count" example.

(a) Explain what the code does in terms of data processing and the final results obtained. Relate your explanation to the concepts of RDD, transformation, and action.

```
someFile = ...
rdd =
    sc.textFile(someFile)\
        .flatMap(lambda line:
            [(word,1) for word in line.split()])\
        .reduceByKey(lambda x,y: x + y) \
        .filter(lambda pair: pair[1] >= 10)\
        .sortByKey(ascending=False)\
        .map(lambda pair: (pair[1], pair[0]))
data = rdd.collect()
```

someFile	Represents the path to the input file
sc.textFile(someFile)	Creates an RDD (Resilient Distributed Dataset) from the text file. **(Transformation)**
.flatMap(lambda line: [(word, 1) for word in line.split()])	Splits each line of the file into words and maps each word to a tuple of (word, 1). **flatMap** is used to flatten the list of tuples into a single list of (word, 1) pairs. **(Transformation)**
.reduceByKey(lambda x, y: x + y)	Groups the tuples by word (key) and sums their counts, effectively counting the occurrences of each word in the file. **(Transformation)**
.filter(lambda pair: pair[1] >= 10)	Filters out words that occur less than 10 times. **(Transformation)**
.sortByKey(ascending=False)	Sorts the word-count pairs by the word (key) in descending order. **(Transformation)**
.map(lambda pair: (pair[1], pair[0]))	Transforms each (word, count) pair into (count, word) pairs, swapping the positions of the count and the word. **(Transformation)**

data = rdd.collect()	Triggers the execution of the transformations and collects the final RDD into a list on the driver program. The result data is a list of (count, word) tuples. **(Action)**

(b) Considering the nature of Spark transformations, which can be narrow ou wide, and the eventual necessity of data reshuffling, explain what Spark execution stages we may have for the processing in the example.

Narrow transformations like flatMap (or map, filter) map one input partition to one output partition.

Wide transformations, like reduceByKey, also called shuffles, may read from several input partitions and contribute to many output partitions. They required data to be reshuffled across executors.

Stage 1: flatMap transformation (Narrow Transformation).
Stage 2: reduceByKey transformation (Shuffle stage).
Stage 3: filter transformation (Narrow Transformation).
Stage 4: sortByKey transformation (Shuffle stage).
Stage 5: map transformation (Narrow Transformation).
The collect action to bring the results to the driver.

Part B

1. Discuss about alternatives for handling **big data** in the Data Sources layer. In other words, what kind of operations, functions and implementations would guarantee a proper handling of data *volume*? Discuss about data loading and data preprocessing.

Data Loading

- Break file in multiple smaller files that can be read in parallel: useful if the reading can be done in parallel
- Undersampling: need to be careful about data distribution
- Use of specific hardware and software: distributed disks, distributed file systems, distributed databases, in-memory databases, parallel and distributed software
- Work with compressed files: zip, parquet, CSR, CSR5 (for sparse matrices) etc

Data Processing

- Data Cleaning
- Normalization and Standardization
- Feature Engeneering
- Dimensionality Reduction
- Sampling

2. With respect to current systems, tailored for big data, studied in class, what are their limitations regarding data characteristics such as velocity, value, veracity and variety?

- **Velocity Limitation:** Big data systems may struggle to handle real-time data processing requirements.
- **Value Limitation:** Big data systems may struggle to extract actionable insights and value from data, especially when dealing with complex analysis requirements or poorly defined business objectives.

- **Veracity Limitation:** Big data systems may face issues with data quality, accuracy, and reliability, especially when dealing with noisy or incomplete data.
- **Variety Limitation:** Big data systems may struggle to handle diverse data types and formats, especially when dealing with unstructured or semi-structured data.

3. Explain the meaning of each line of Python code below:

```python
def run(model_dir, feature_extraction, sink, beam_options=None):
    print('Listening...')
    with beam.Pipeline(options=beam_options) as p:
        _ = (p
            | 'Feature extraction' >> feature_extraction
            | 'Predict' >> beam.ParDo(Predict(model_dir, 'ID'))
            | 'Format as JSON' >> beam.Map(json.dumps)
            | 'Write predictions' >> sink)
```

def run(model_dir, feature_extraction, sink, beam_options=None):	Defines a function named **run** that takes four arguments: **model_dir**, **feature_extraction**, **sink**, and optionally **beam_options**. The function is responsible for running a data processing pipeline.
print('Listening...'):	Prints the message "Listening..." to the console. It serves as a notification that the program is actively waiting for incoming data.
with beam.Pipeline(options=beam_options) as p:	Starts a new data processing pipeline using Apache Beam. It creates a pipeline object named **p** and utilizes the provided **beam_options** for configuration. The **with** statement ensures that resources associated with the pipeline are properly managed.
_ = (p	Initializes a new step in the data processing pipeline. _ is used to capture the output of the pipeline.
|	Applies the PTransform on the right side of the pipe to the input PCollection.
>>	Add a label to the transforms, like 'Feature extraction' >> feature_extraction.
| 'Feature extraction' >> feature_extraction	Defines a step in the pipeline for feature extraction.
| 'Predict' >> beam.ParDo(Predict(model_dir, 'ID'))	Defines a step in the pipeline for prediction.
| 'Format as JSON' >> beam.Map(json.dumps)	Defines a step in the pipeline for formatting the predictions as JSON.

| 'Write predictions' >> sink)	Defines a step in the pipeline for writing the predictions to a sink.

4. Give two advantages and two disadvantages of the utilization of GPUs for data processing and analysis.

Advantages

- **Parallelism:** GPUs are designed with a large number of cores that can perform many operations in parallel, which significantly speeds up data processing tasks that can be parallelized
- **Performance:** For specific tasks, such as those found in machine learning and scientific computing, GPUs can outperform CPUs by a substantial margin.

Disadvantages

- **Program Complexity:** Utilizing GPUs effectively often requires specific programming models and libraries like CUDA or OpenCL. This adds complexity compared to traditional CPU programming, demanding specialized skills and potentially increasing development time.
- **Limited General-Purpose Functionality:** While powerful for specific tasks, GPUs are not replacements for CPUs. They struggle with non-parallelizable tasks and general-purpose computing. A system typically requires both CPUs and GPUs, increasing hardware complexity and potentially software development overhead.

5. Given the Python code below, explain the function of lines: 6, 7, 10, 12, 14 and 16.

```python
1   def parallel_map(function, iterable):
2       if FORCE_DISABLE_MULTIPROCESSING:
3           return [function(*args) for args in iterable]
4
5   original_sigint_handler = signal.signal(signal.SIGINT, signal.SIG_IGN)
6   num_threads = mp.cpu_count() * 2
7   pool = mp.Pool(processes=num_threads)
8   signal.signal(signal.SIGINT, original_sigint_handler)
9
10  p = pool.map_async(_function_wrapper, ((function, args) for args in iterable))
11  try:
12      results = p.get(0xFFFFFFFF)
13  except KeyboardInterrupt:
14      pool.terminate()
15      raise
16  pool.close()
17  return results
```

- **Line 6:** Calculates the number of processes to be used in the multiprocessing pool.
- **Line 7:** Creates a pool with the specified number of worker processes.
- **Line 10:** asynchronously maps the provided function over the iterable using the pool

- **Line 12:** Retrieves the results from the asynchronous map operation. 0xFFFFFFFF represents a large timeout value, effectively waiting indefinitely until all the results are available.
- **Line 14:** Terminates the pool (stop all worker processes) in case of an exception
- **Line 16:** Closes the pool. Ensures proper resource management, avoids potential leaks and prevents any more tasks from being submitted to the pool.

6. Figure 1 presents a set of data for binary classification. The problem is to build a predictive model that discriminates trains that go east from trains that go west. Represent this problem with a single bidimensional table. What are the disadvantages of representing this problem in this format? (Note: this problem could be solved using image processing machine learning algorithms, but we are interested in "interpretable" and "explainable" models.)

- Build a table with the number of each type of carriage
- Disadvantage: lose the way each carriage is ordered

7. Given the slice of code below, what would be the problem of allocating multiple threads to execute this loop?

```
do I=2,9
    X[I] = Y[I] + Z[I]
    A[I] = X[I-1] + 1
enddo
```

The calculation of A[I] in each iteration depends on the value of X[I-1] from the previous iteration. This creates a sequential dependency, where each iteration must wait for the previous iteration to complete before it can proceed, and without additional synchronization mechanisms it would result in incorrect computations.

8. Explain the differences between the two script codes (CODE 1 and CODE 2) below. **Do not omit details.**

CODE 1

```
WORK_DIR=/tmp/cloudml-
samples/molecules
python predict.py \
  --work-dir $WORK_DIR \
  --model-dir $MODEL_DIR \
  batch \
  --inputs-dir $WORK_DIR/data \
  --outputs-dir
$WORK_DIR/predictions
```

CODE 2

```
PROJECT=$(gcloud config get-value
project)
WORK_DIR=/tmp/cloudml-
samples/molecules
python predict.py \
  --work-dir $WORK_DIR \
  --model-dir $MODEL_DIR \
  stream \
  --project $PROJECT \
  --inputs-topic molecules-inputs \
  --outputs-topic molecules-
predictions
```

- **WORK_DIR:** Defines the working directory.
- **python predict.py:** Executes the predict.py script.
- **--work-dir $WORK_DIR:** Specifies the working directory for the script
- **--model-dir $MODEL_DIR:** Specifies the directory containing the model.

Differences

- **CODE 1**
    - **batch:** Indicates batch processing mode. Used for processing a large set of inputs in a single run.
    - **--inputs-dir $WORK_DIR/data:** Directory for input data.
    - **--outputs-dir $WORK_DIR/predictions:** Directory for output predictions.
- **CODE 2**
    - **stream:** Indicates streaming processing mode. Used for real-time data processing.
    - **PROJECT=$(gcloud config get-value project):** Fetches the Google Cloud project ID.
    - **--project $PROJECT:** Specifies the Google Cloud project ID. This is necessary for interacting with Google Cloud services.
    - **--inputs-topic molecules-inputs:** Input topic from which the script will read data. This is typical for streaming data from a messaging system like Google Cloud Pub/Sub.
    - **--outputs-topic molecules-predictions:** Output topic to which the script will write predictions.