Big Data Technologies 774/874

Assignment 2

Overview

Part 1 Goals

- Revisit the concepts of the lectures and apply some of the information/lessons to *new situations*.

Part 2 Goals

- Consider the Map Reduce approach in more detail.

Submission

- Answers to be submitted on SUNLearn.

Part 1: Big Data Concepts

Question 1: Big Data General [7]

1. From an enterprise's point of view, which V is the most important? Motivate your answer. [2]
2. Which V would you consider to the be the next most important in a general enterprise setting. Motivate your answer. [2]
3. Identify key drivers of big data adoption in enterprises and motivate your choices. [3]

Question 2: Data Lifecycle [7] In the lectures we discussed the concepts of DevOps, DataOps and MLOps. These approaches allow us to deliver code, data or models quickly and reliably.

1. Describe the principles governing these approaches. In your response, specifically compare MLOps and DevOps (e.g. how do they approach CICD, testing, teams, etc)? [5]
2. Why are these approaches of particular importance in big data? [2]

Question 3: Data Storage Architectures [9] You are currently utilising a data warehouse within your organisation in the construction of data science models. There has been a drive to move to big data technologies (in this case Hadoop) and one of your use-cases has been selected for the migration. The organisation is taking the approach of running a proof-of-concept (POC) with a few on-premises Hadoop machines. You have joined a team that is planning this POC. Your team will own the data lifecycle from ingestion and storage, through to using these data for modelling. This POC is a precursor to a data lake.

1. Describe the differences and similarities between data warehouses and data lakes, as well as when to use each data architecture? [5]
2. Let's suppose your model (M) relies on two tables (t1, t2) in the data warehouse, each of which are directly dependent on two operational/source systems (say t1(s1, s2), t2(s3, s4)). Describe, in high-level terms, how the necessary data should be ingested into the data lake (in terms of what is retrieved and how it is written). Lastly, what processing/ingestion technology would you use with your on-premises Hadoop infrastructure? [2]
3. Once the data has arrived in the lake, what will you need to do to prepare it for use by your model? How would the dependencies of M change and what additional work would need to be done to make your model pipeline compatible? [2]

Question 4: Sharding and Model Serving [8] Suppose you have produced a simple prediction model that has been containerised and deployed on infrastructure like Kubernetes (K8S), configured to autoscale your service. As part of your model lifecycle, you wish to capture all predictions made when users interact with the service. You are currently storing these data to a sharded NoSQL technology (say MongoDB for the sake of this question), and are using range partitioning on the timestamp to distribute your data.

1. Describe what sharding is, and describe range sharding in particular. [2]
2. What problems/issues is sharding solving? [3]
3. What happens if your service gains in popularity? Is this sharding solution still viable? [3]

Question 5: Distributed Filesystems [8] Consider HDFS and Google's distributed file system (GFS — in the form discussed in the paper by Dean et al. and presented in the lectures).

1. How is fault tolerance achieved and what are the typical failures that these distributed filesystems are mitigating? [3]
2. What kind of replication approach do these systems use insofar as it relates to peer-to-peer or master-worker architectures. How are replicas distributed and how are data located within this filesystems? [3]
3. Why is it useful to implement large, fixed chunk/file sizes in GFS & HDFS? [2]

Part 2: Map Reduce

Question 6: Minimal Map Reduce [9] Suppose we wish to count the number of characters in text files on a system of three nodes (Alpha, Beta, Gamma), and we wish to use the Map Reduce paradigm. For the sake of this example, we will assume the following distribution of the data across the nodes:

Node	File Contents
Alpha	abcdabcdabcd
Beta	axyaxyaxy
Gamma	adyadyady

1. Write a pseudo code implementation of a Map and Reduce function to calculate the character count (see Dean et al. for inspiration). [3]
2. Provide the output of the *map* tasks and how the output is distributed over nodes (e.g. Node1: (key1, value1), (key2, value2), Node2: (key3, value3)…). [2]
3. Suppose you have two reducer processes (say, R1 and R2) that will execute your custom reducer code, which data will be processed at each reducer (e.g. R1: (key2, value2), R2: (key3, value3), (key1, value1)))? [2]
4. During this entire process, which data are written to disk? [2]
5. You wish to reduce network traffic between the mappers and the reducers using an additional stage, what is this stage called and how would it act on the data in this example? Demonstrate this by modifying the key-value pairs as necessary. [2]

For the master's level, Big Data Technology (Eng) 874 (not 774), students

Question 7: An Inverted Index [18] Consider the paper on map reduce (Dean et al.). We wish to implement a solution to the simple inverted index problem described there:

> InvertedIndex: The map function parses each document, and emits a sequence of `<word; document ID>` pairs. The reduce function accepts all pairs for a given word, sorts the corresponding documentIDs and emits a `<word; list<documentID>>` pair. The set of all output pairs forms a simple inverted index. It is easy to augment this computation to keep track of word positions.

For example if we have two documents:

Label	Contents
D1	the cat sat on the mat
D2	the dog sat on the log

our simple inverted index would be

Word	Document(s)
the	D1, D2
cat	D1
dog	D1
sat	D1, D2
on	D1, D2
mat	D1
log	D1

By precomputing these indices we can drastically improve document retrieval given search terms.

1. Use a map reduce programming approach (in a language or tool of your choice) to compute a simple inverted index using the example above. [5]
2. Apply this approach to three tragedies by William Shakespeare, namely Romeo and Juliet, King Lear and MacBeth. Perform some preprocessing of the data (e.g. remove stop-words) and provide an excerpt of twenty words from your index. You may use the URL as document identifier (use your implementation from question 1). [5]
3. Expand the approach to compute `(word, [(URL, position)])`, and order by document identifier (URL), then by word position. Positions in this case are word locations within the document (e.g. for D1; `(the, 0), (cat, 1), (sat, 2), ..., (mat, 5)`). Newlines should be ignored and you should consistently apply the same data cleansing steps. Submit your source code and twenty words from your index. [8]

Hints: An example implementation of *word count* is provided that utilises the command line (with python installed) here — you may modify this solution or use Apache Beam. Furthermore, the NLTK in python is useful for preprocessing/preparing text.

For interest's sake, explore Elastic Search after this assignment to see how they manage full-text search (with an inverted index beneath).