CMI/Jan-Apr-2020/End-Semester Exam Aug 2020

Chennai Mathematical Institute

Big Data with Hadoop Deadline: Section I: 1 PM Section II: 5 PM.
Max Marks: 35 (for Section I) + 20 (for Section II).

Roll No.: _____ Date: Aug 2020

Name: _____

Instructions

- This is an open book test.
- Calculators are allowed.
- Please do not mail your answers to your instructor. Upload your answers (as a single PDF file for each Section) to Moodle. Name your pdf as <rollno>-<fname>-sec1.pdf and <rollno>-<fname>-sec2.pdf. If the upload fails, mail your work to venkateshv@iiitd.ac.in with subject line "BDH Exam".
- You have 4 hours to complete Section 1. You have an additional 4 hours to complete Section 2.
- No negative marks for any section.
- You may use internet resources. Do not discuss with friends. Do not engage in any kind of plagiarism.

Section 1: Questions 1 - 5 carry 7 mark each.

Question 1. Draw a sequence diagram to explain the process of updating free space bitmap. In your own words, explain the sequence diagram.

Solution A sequence diagram depicts the interaction between instances of objects in a system. Here, we consider the interactions between the user and the disk through the file system. Finding suitable abstractions helps us manage the complexity of interactions. There are many ways to represent the interactions. Here is one way where we use four instances as suitable abstractions for drawing.

When the user (an instance of whom is represented as *:User*) writes a block, the operating system (represented as *:OS* through the file system component) needs a free block address to place the block on the disk (represented as *:Disk*). Let us say, this component is called the Free Space Manager referred to as *FreeSpaceMgr* in Figure 1. The Free Space Manager need to access the bitmap from the disk, update it to reflect that this block is now updated with the to-be written block. Once it has the address, it may write to the new block. This is just one part of the sequence diagram. Similarly, deletion of a block can be represented in a sequence diagram.

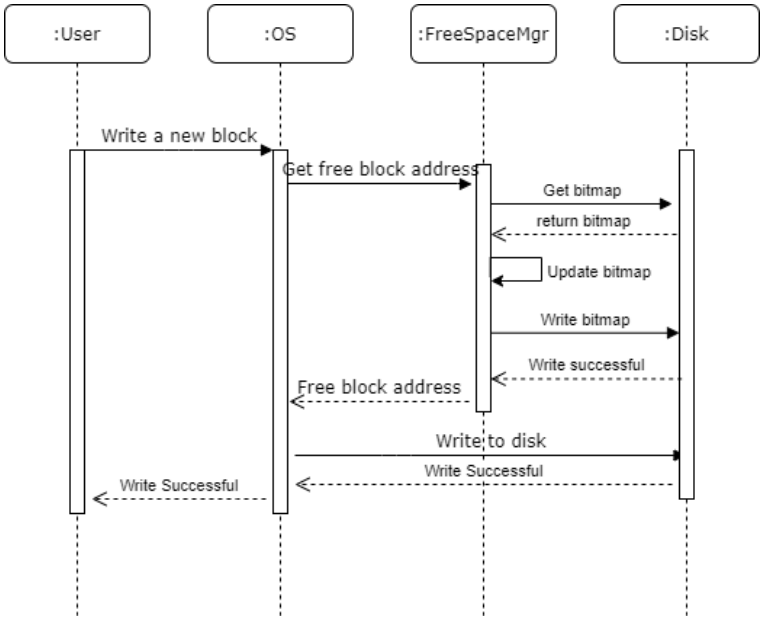

FIGURE 1. Sequence Diagram

Question 2. A distributed computing environment uses global vector time stamps as discussed in the class. You are provided with the following facts:

(1) $(1,1,1,1) \rightarrow (2,1,1,1)$ is a happens-before relation.
(2) Exactly two events occurred in each process.
(3) Between the two events of every process, at least one event occurred in another process.
(4) If the last digit of your roll number is r, the first event occurred in the process p_i where $i = (r\%4) + 1$.

Draw the hasse diagram and list the global vector time stamps. Note that there may be more than one hasse diagram agreeing to the above facts. You may draw any one.

Solution

The dimensions of the happens-before vector gives away the number of processes. With four processes, Figure 2 gives one possible interaction between our distributed processes. Note that we follow a global vector time stamps here. Depending on your roll number, instead of p_1, the first event will occur somewhere else. Also, note that the $(1,1,1,1) \rightarrow (2,1,1,1)$ is a happens-before relation in the diagram. The second event happens first on p_1.

In our construction, we have a total order. Therefore, the hasse diagram is just a straight line as dictated by the timestamps.

Question 3. Given any vector $z = (z_1, z_2, ...z_K) \in \mathbb{R}^K$, its distinctly valued components (z_i) may not add up to 1. There are many ways to transform z into another vector such that this property is achieved. Here are two ways:

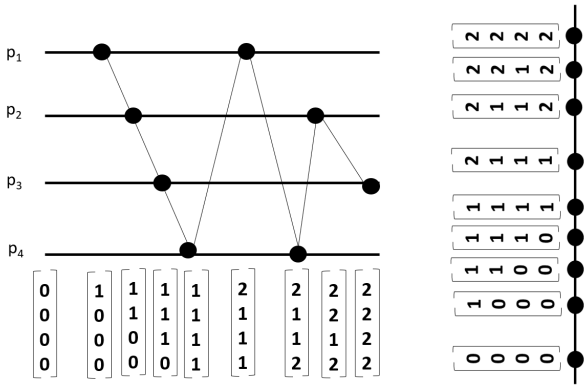

FIGURE 2. Global Vector Time Stamped on Processes and the Corresponding Total Order of Events

(1) Use $argmax(z)$ where the component with maximum value gets 1 and all others get 0. For example if $z = (1, 2)$, our new representation $argmax(z) = (0, 1)$.
(2) We can use softmax function, $\sigma(z) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$.

Assuming we have a large number of such fixed length vectors, provide map-reduce design(s) to calculate both the argmax and softmax values. The input vectors are presented as a HDFS file with each line containing a vector. You are required to get the output stored as two separate files, one for argmax vectors and the other for softmax vectors. You need not code. Draw and explain the map-reduce design.

Solution In the simplest design, a mapper is sufficient to convert a vector to its argmax and softmax representation. The same mapper can write two files as output. This is because we don't need more than one line from input to arrive at the output. Figure 3 shows the idea pictorially. A "Filter" is not a suitable term here as we don't filter any row. We just process it.

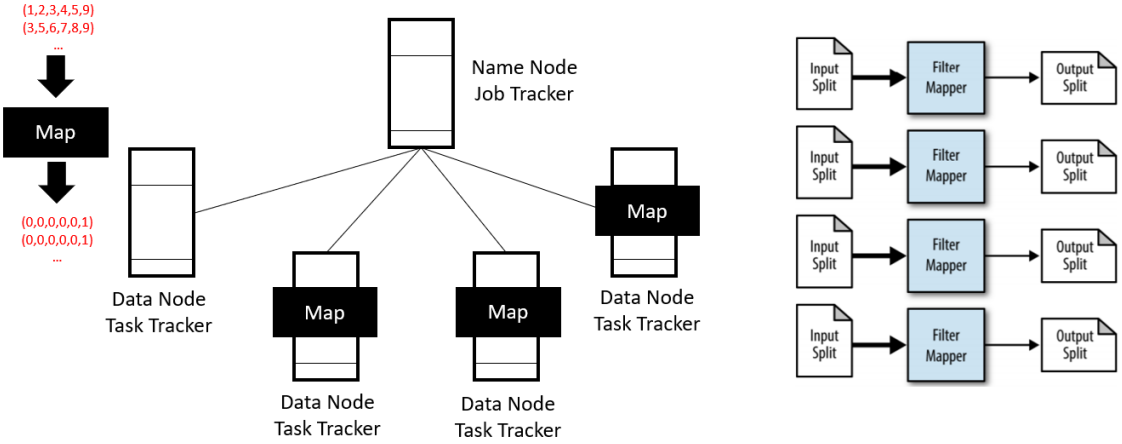

FIGURE 3. A Simple Map Reduce Pattern

Question 4. You are hired as a DB administrator for Chunnalal Travel Agency (CTA). CTA is building a ticket reservation web service. CTA faces the following challenges and looks up to you for suggestions.

(1) CTA recently recruited staff skilled in key-value datastores. Identify two scenarios where a key-value datastore can by used by CTA.
(2) If every ticket ever booked through the service is logged for future reference, which type of datastore will you use and why?

Solution Key-Value stores have a reputation of being very fast for reads and writes. Another advantage is that the storage overhead is minimal. There are more advantages. Yet, these two lead to the following use-cases:

(1) Real-time processing: CTA can introduce a real-time "number of users logged in" ticker to show that their site is popular.
(2) Caching to enable other web services: As we saw that redis is used in the load balancing use case, CTA could store information such as "trains running discounted fare" for search related to reservation.

Several kinds of log formats, the volume of logs and the write-once nature of logs make NoSQL datastores perfect candidates for their storage. We discussed, key-value, graphdb, document db and columnar db in class. If logs can be structured such as say, *ID, timestamp, log string, Additional Data*, then we could consider a columnar db. As additional data grows or becomes more structured, columnar db gives us an advantage of querying them effectively. We also saw JSON based semi-structured datastores that could also serve the same purpose. Here, it is important to understand if we prefer consistency, partition-tolerance or availability over one other. We can achieve two of the three. Since logs are not sensitive for consistency, a datastore that prioritize for AP such as Cassandra or CouchDB would be more appropriate.

Question 5. CMI wants to build a web service which provides a knowledge repository such as Wikipedia for explaining the concepts of algebraic geometry. You may select any small part of this system (for example, adding a concept, accepting comments, etc) to illustrate how to design a web service. Design a REST based web service for this purpose and through that scenario, explain the role of non-idempotent REST methods.

Solution Designing a REST service involves four key steps which we detail here. This solution hint considers "Adding a concept" scenario. It can be extended for any such scenarios.

(1) Identify the object model: concept
(2) Create Model URIs: /concept/{ID}
(3) Determine Representations: An XML representation is shown below.
(4) Assign HTTP Methods: POST is a non-idempotent method. A post call will change the state of the system. We use it here to create a new post as in "HTTP POST /concept". This takes the representation XML of concept as data payload. Server creates the concept on receiving a post request. It could create an entry in its database and return a "HTTP 200 OK" response if everything is successful.

A sample representation of a concept:

```
<concept id=1>
    <name>Algebraic Variety</name>
    <description>
    An algebraic variety is defined as the set of
    solutions of a system of polynomial equations over the real
    or complex numbers.
    </description>
</concept>
```

Section 2: Questions 6 - 9 carry 5 mark each.

Question 6. Some students at CMI wanted to publish the courses offered in a particular semester, its pre-requisite courses, the instructor, teaching assistant and such details as a neo4j graph. Can you setup a part of this graph using neo4j sandbox/desktop? Provide the neo4j graph, the neo4j commands (and a sample of data if any) that would create such a graph. Your graph must have at least 7 nodes and 12 nodes as the maximum.

Solution The graph shown in Figure 4 can be generated using the following commands:

```
//Lets clean up everything.
match (n) detach delete n;

//Setup the graph. A semicolon helps you to run multiple queries together. If
     this does not work, enable the settings appropriately.

//Creating pre-requisites
create (c1:Course{name:'DMML'})-[:IS_A_PREREQUISITE_OF]->(c2:Course{name:'AML
    '});
match(c2:Course{name:'AML'}) create (c1:Course{name:'NLA'})-[:
    IS_A_PREREQUISITE_OF]->(c2);
match(c2:Course{name:'NLA'}) create (c1:Course{name:'LAA'})-[:
    IS_A_PREREQUISITE_OF]->(c2);

//Creating teaches
match (c2:Course{name:'DMML'}) match (c1:Course{name:'AML'}) create (a:
    Instructor{name:'Madhavan'})-[:TEACHES]->(c2) create (a)-[:TEACHES]->(c1)
    ;
match (c2:Course{name:'NLA'}) match (c1:Course{name:'LAA'}) create (a:
    Instructor{name:'Kavita'})-[:TEACHES]->(c2) create (a)-[:TEACHES]->(c1);

//Creating offered-in
```

```
match (c2:Course{name:'NLA'}) match (c1:Course{name:'LAA'}) create (c2)-[:
    IS_OFFERED_IN]->(a:Semester{name:'Sem2'}) create (c1)-[:IS_OFFERED_IN]->(
    b:Semester{name:'Sem1'});
match (c2:Course{name:'DMML'}) match (c1:Course{name:'AML'}) match (e:
    Semester{name:'Sem2'}) create (c2)-[:IS_OFFERED_IN]->(e) create (c1)-[:
    IS_OFFERED_IN]->(b:Semester{name:'Sem3'});

//Creating is_ta_for
match (c2:Course{name:'NLA'}) match (c1:Course{name:'LAA'}) create (a:
    Assistant{name:'Abhishek'})-[:IS_TA_FOR]->(c1) create (b:Assistant{name:'
    Anubhab'})-[:IS_TA_FOR]->(c2);
match (c2:Course{name:'DMML'}) match (c1:Course{name:'AML'}) create (a:
    Assistant{name:'Debjit'})-[:IS_TA_FOR]->(c2) create (a)-[:IS_TA_FOR]->(c1
    );

//See the graph
match (n) return n;
```

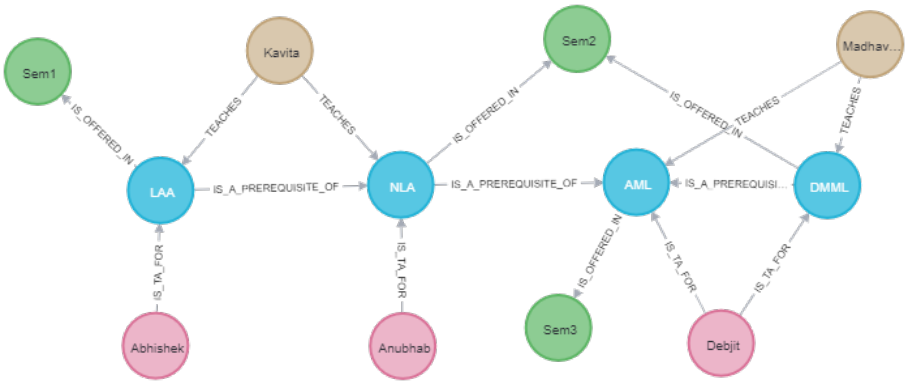

FIGURE 4. Neo4j Graph

Question 7. Design an Apache Storm topology for softmax and argmax computation as explained in Question 3. For this purpose, draw the topology diagram and explain the streams, spouts and bolts. You do not need to write any code.

Solution Typically storm topologies are used to setup realtime computation. Here, let us assume that the conversion is required to support some other realtime application. Another possible usecase is that the HDFS stores are different.

In a storm topology, Stream is an unbounded sequence of tuples. Spouts create the streams. Spouts connect to the source such as web servers or HDFS (as in this case) to pull data. Here, we have only one kind of intput. But, it requires two different kinds of processing. Hence, we create two bolts, one each to process argmax and sosftmax respectively. Technically, it is possible to have the same bolt do both. The separation helps us in future-proofing our design, just in case, we extend these ideas into many more complex

computations. Bolts process the streams and emit new streams. Since we need all records from spout to go to both the bolts, we use "All" grouping.

Spouts and Bolts run in parallel. Storm guarantees no-data-loss even if some nodes go down while processing. Moreover, it allows us to implement flow-control. With flow-control, we can decide the maximum data that can be queued up as tuples on stream for a bolt. Figure 5 shows two topologies, one with a single HDFS Bolt and all-grouping, and another topology with multiple HDFSBolts with shuffle-grouping.

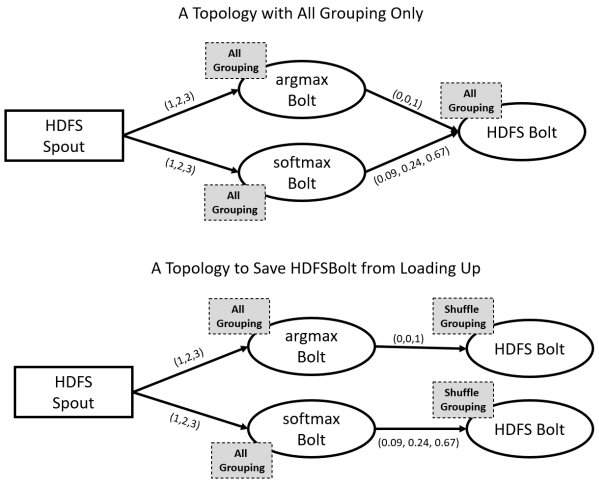

FIGURE 5. Two topologies with different grouping strategies.

Question 8. GRE and GMAT exams are conducted in computer-based testing formats. Students visit individual testing centers in their local town to take the exam. As and when the student completes an exam, each system in this testing center submits the test data to either of the two web services namely the GRE and GMAT web services. These web services not only capture the test set details, the student solutions, but also capture the other data such as time spent by each student on each question, the number of times a particular question was revisited and the order of answering the questions. We wish to save these data in two different HDFS stores, a) one for storing the results, b) another for rest of the data for research on how to improve the tests.

(1) Draw and explain a multi-agent data flow model with flume for the above-mentioned scenario. (3 marks)
(2) Within any part of the given scenario, identify a flume event and explain its contents with examples. (2 marks)

Solution Given that there are two HDFS stores to save the results to, we configure two sinks. Similarly we have two sources subscribed to two different data elements. So, a simple multi-agent flume data flow model will look like Figure 2;

The role of flume source is to receive events and store them in one or more channels. Events stay in channels until flume knows that the event is moved by the sink successfully.

Sink removes the event from channel after its job is over. So, the channel capacity plays a major role in the reliability of the system.

Figure 6 shows a flume multi-agent data flow model.

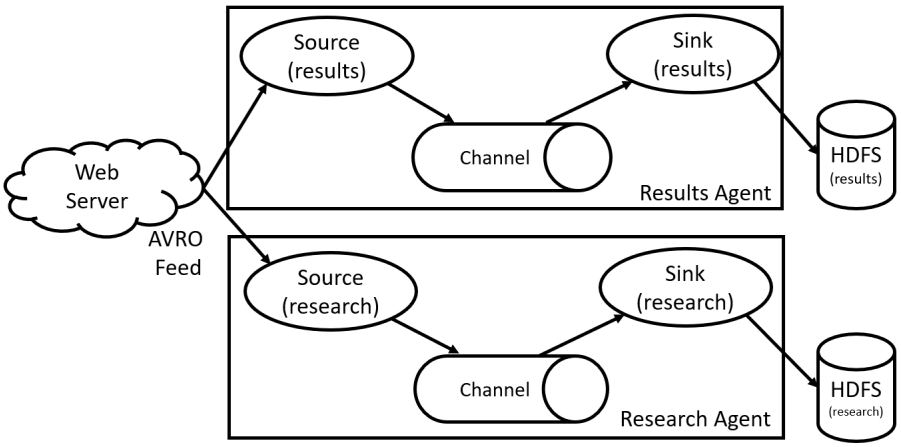

FIGURE 6. Flume Data Flow Model

A flume event is the unit of data flow containing a payload and optional attributes. We could have the test results published as a Flume Event as follows: TestTakerID, Score. As a concrete example, data records may flow as follows: {(VEN1000202, 98), (MMD000203, 87), ... }. Also, assume some attributes are set in the input to distinguish between results and the rest. Say, an attribute "type" has a value "TestScore", then our results agent is interested in it. This can be represented in a JSON format for which an AVRO schema can be created as shown below.

```
{
    "type": "string",
    "TestTakerID": "string",
    "score": "int"
}
```

Once the schema is available, this data can be serialized in binary format and used by flume. Both the data producer and the flume (consumer) must have the same AVRO schema to work with the data feed.

Question 9. CMI was excited to go digital by streaming its annual day festival events. Specifically, it published the results of all the events in real-time. Three news TV channels decided to subscribe to these events and show them as running news repeatedly throughout the day. Draw a kafka-based publish subscribe architecture to implement this use-case. Take any example within the scope of the given use-case to explain *topics* and *partitions*.

Solution Some key facts to know about kafka[1] are:

[1]See https://kafka.apache.org/intro for more information.

(1) If a producer publishes two events with same topic and to the same partition, Kafka ensures that any consumer reading these events will see the events in the same order.
(2) Events are organized and stored as topics.
(3) Every new event published is appended to one of the topic's partitions.
(4) If the publisher does not specify the partition ID, the message key is used to decide the partition.

Let us assume that CMI festival publishes results of on-going chess and coding competitions. We have one producer publishing events to two topics namely chess and coding competition. There are three consumers, the three TV channels. Lets call them SunTV, StarTV and MoonTV. To improve performance, you may increase the number of Kafka brokers (servers that store the partitions are called brokers) as and when necessary. So, the architecture looks as shown in Figure 7.

All CMI writes are organized and stored into the two topics. Each topic is configured to have p partitions. In this example, CMI writes pairings and results data for chess competition with these keys. So, with a configuration of two partitions for the Chess topic, kafka assigns each key to a partitions. Since the partitions may reside in different servers (brokers), this allows for parallelism. Increasing the partitions improves throughput because the consumers can access them in parallel.

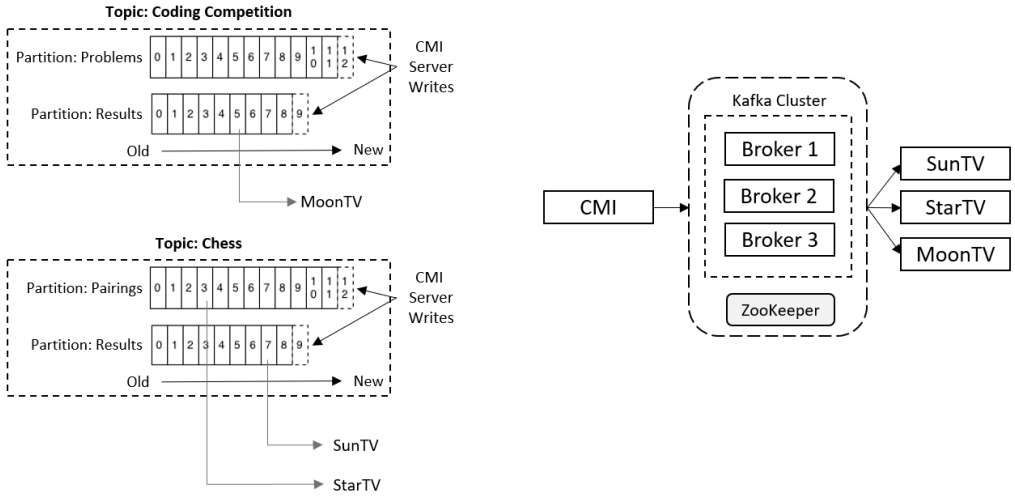

FIGURE 7. Kafka Architecture for CMI Events