CS451 Sample Exam Question Solutions

Sections 2+3

In MapReduce, how does the framework determine which reducer should receive a given intermediate result?

a) Mappers exchange messages to determine the reducer for each key
b) Reducers exchange messages to determine the reducer for each key
d) Keys are assigned to reducers in a round-robin manner
e) Key-value pairs are assigned to reducers in a round-robin manner
f) Mappers use a hash function on each key to determine the reducer for that key

In a MapReduce program, what are the benefits and pitfalls of using an IMC (In-Mapper Combiner) instead of a traditional Combiner?

Benefits: By aggregating data in memory before emitting results, IMC can reduce the total number of emitted intermediate results. This decreases the number of messages between mappers and reducers, reducing both network traffic and disk writes.

Pitfalls: IMC requires sufficient memory to store all intermediate results until the mapper has completely processed all key-value pairs in an input split. Since IMC requires preservation of state between calls to map, it requires careful implementation to avoid bugs, as well as setup and cleanup methods on the mapper.

Consider the following Spark code:

```
def noisySquare(x : Int) = {
  print(f"I'm Squarin' $x")
  x*x
}

def mystery(n : Int) = {
  sc.parallelize(1 until n).map(noisySquare)
```

}

a) What is the type of the RDD returned by the mystery function?

The mystery function returns an RDD[Int]. The mystery function creates a parallel collection using sc.parallelize, which generates an RDD[Int]. The noisySquare function returns an Int, so applying the map transformation returns an RDD[Int]

b) What will be printed to the screen if I have the assignment val x = mystery(3)? (Ignoring any Spark logging output, of course)

Nothing will be printed, since there is no action called on the RDD. This is due to Spark's lazy evaluation mechanism. To execute the transformations, we would need to call an action on the RDD, such as adding x.collect().

Section 4

Describe why we would want to "smooth" our n-grams when training a language model.

Smoothing removes zeros, or discontinuities, from the probability distributions. These zero probabilities would make any sentence containing an unseen n-gram impossible according to the model, even if the sentence is valid otherwise.

True or False: If a posting list is in sorted order, then VInt compression will always resulting in a smaller posting. Justify your answer (if your answer is "false" you can justify it by counterexample).

False. If the gaps between each value in the posting is very large, say 2^28, storing the integer values themselves will take only 4 bytes each, but storing as VInts will take 5 bytes each, resulting in a larger posting.

Section 5

When performing single-source-shortest-path in MapReduce / Spark, we used Breadth-First-Search instead of Dijkstra. Why is that?

Dijkstra's algorithm cannot be effectively parallelized, since it relies on a priority queue where vertices are removed one at a time.

Why are spider-traps a problem in Page Rank? How is this problem resolved?

In a spider-trap, all pages within the group only link to each other. As PageRank iterates, all importance gets absorbed by the spider-trap, resulting in those pages receiving artificially high rankings while other pages get zero or very low rankings. In the random surfer model, we can view it as the surfer being trapped in the spider-trap.

We can resolve this by adding random teleportation: when visiting a page, we either follow an out-link at random with probability β, or jump to a random page in the network with probability (1- β). This ensures that after a finite number of steps, the surfer will eventually teleport out of the trap.

Why are dead-end nodes a problem in Page Rank? How is this problem resolved?

Since a dead-end has no out-links, during PageRank iteration it will leak mass out of the system. This is because dead ends make the matrix no longer column-stochastic.

We can resolve this by always teleporting at a dead end, to a random page. In MapReduce, this can be done by replacing dead-ends to links to everyone, or redistributing the missing mass back in post-processing.

Section 6

If doing machine-learning on user data, is age a sparse or a dense feature?

Age is a dense feature. The value will almost always be non-zero.

Assume we have k centroids and a dataset in x HDFS blocks. When running K-Means Clustering, what is the maximum number of:

Mappers: x

Reducers: k

Explain your reasoning.

Since the dataset is in x HDFS blocks, we can use x Mappers to process them; they can assign points to clusters in parallel. Each reducer recomputes the centroid of one cluster. Since the points are partitioned by cluster, this allows us to efficiently complete the computations.

Section 7

When is it appropriate to perform a Hash Join (also known as a Broadcast Join)? When is it inappropriate?

A hash join can be used when one of the sets is small enough to fit into the memory of a node.

Given a DataFrame people with 3 columns: name: String, age: Integer, job: String

Compute the average age for each unique job. You can use Spark SQL or DataFrame operations. The DataFrame API can be found on the handout. You can assume the DataFrame has already been registered as a View with the name people. (Note: We didn't actually do this on an assignment so I'd imagine most students would pick SQL? But it's borderline if I'd want to ask you)

Spark SQL

```
val result = spark.sql(“
      SELECT job, AVG(age) FROM people
      GROUP BY job
“)
```

DataFrame API

```
val result = people.groupBy("job").agg(avg("age"))
```

Section 8

True / False: Spark Streaming processes incoming messages as soon as they arrive. Explain your answer.

False. Spark Streaming (DStream) collects data every t seconds (configurable, usually t=1) into an RDD, creating a stream of RDDs.

If I have a DStream ds, and create a new DStream newDS = ds.flatMap(f) which of the following is true?

a) There will be a 1:1 matching between RDD in newDS and ds
b) Every RDD in ds will map to 1 or more RDD in newDS
c) Every RDD in ds will map to 0 or nore RDD in newDS
d) There is no correlation between RDD in ds and newDS

Assume we have an DStream[String] called "words" and we want, every minute, to print the LONGEST word seen in the last minute. (Ties can be broken in any way you like) Complete the following code by giving definitions for f1, f2, x, and y.

(Note: The ByWindow / AndWindow functions don't NEED a "inverse" function, they just aren't as efficient if you can't come up with one. With things like min/max it's not really possible to reverse efficiently, so we must deal)

```
val f1 = (word1: String, word2: String) => if (word1.length >= word2.length)
word1 else word2

val f2 = (rdd: RDD[String]) => {
  if (!rdd.isEmpty()) {
    val longestWord = rdd.first()
    println(s"Longest word in the last minute: $longestWord")
  }
}

val x = Minutes(1)

val y = Minutes(1)

words.reduceByWindow(f1, x, y).foreachRDD(f2)
```

Section 9

Bloom filters:

a) Can give false positives (falsely report a value has been seen)
b) Can give false negatives (falsely report a value has not been seen)
c) All of the above
d) None of the above

In Redis, keys are hashed and taken modulo 16384. Each of these 16K buckets is assigned to a single server, and transactions can only insert / delete / update a single key at a time. The cluster controller is in charge of assigning buckets to nodes in the cluster. If the controller loses contact with a node, it will reassign the buckets to another node (using replicas). If a node loses contact with the controller, it will go into offline mode and will not accept transactions until it reestablishes connection. In terms of the CAP theorem, what category does Redis fall under? In terms of PACELC, what category does Redis fall under?

CAP

a) PC
b) PA
c) CA

PACELC

a) PA/EL
b) PA/EC
c) PC/EL
d) PC/EC