
# Free Questions for HDPCD by certscare

## Shared by Rosario on 20-10-2022

**For More Free Questions and Preparation Resources**

**Check the Links on Last Page**

# Question 1

**Question Type:** MultipleChoice

You want to perform analysis on a large collection of images. You want to store this data in HDFS and process it with MapReduce but you also want to give your data analysts and data scientists the ability to process the data directly from HDFS with an interpreted high-level programming language like Python. Which format should you use to store this data in HDFS?

## Options:

**A-** SequenceFiles

**B-** Avro

**C-** JSON

**D-** HTML

**E-** XML

**F-** CSV

## Answer:

B

# Question 2

**Question Type:** MultipleChoice

When can a reduce class also serve as a combiner without affecting the output of a MapReduce program?

## Options:

**A-** When the types of the reduce operation's input key and input value match the types of the reducer's output key and output value and when the reduce operation is both communicative and associative.

**B-** When the signature of the reduce method matches the signature of the combine method.

**C-** Always. Code can be reused in Java since it is a polymorphic object-oriented programming language.

**D-** Always. The point of a combiner is to serve as a mini-reducer directly after the map phase to increase performance.

**E-** Never. Combiners and reducers must be implemented separately because they serve different purposes.

## Answer:

A

## Explanation:

You can use your reducer code as a combiner if the operation performed is commutative and associative.

# Question 3

**Question Type:** MultipleChoice

Which best describes what the map method accepts and emits?

## Options:

**A-** It accepts a single key-value pair as input and emits a single key and list of corresponding values as output.

**B-** It accepts a single key-value pairs as input and can emit only one key-value pair as output.

**C-** It accepts a list key-value pairs as input and can emit only one key-value pair as output.

**D-** It accepts a single key-value pairs as input and can emit any number of key-value pair as output, including zero.

## Answer:

D

## Explanation:

public class Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT>

extends Object

Maps input key/value pairs to a set of intermediate key/value pairs.

Maps are the individual tasks which transform input records into a intermediate records. The transformed intermediate records need not be of the same type as the input records. A given input pair may map to zero or many output pairs.

Class Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT>

# Question 4

**Question Type:** **MultipleChoice**

Workflows expressed in Oozie can contain:

## Options:

**A-** Sequences of MapReduce and Pig. These sequences can be combined with other actions including forks, decision points, and path joins.

**B-** Sequences of MapReduce job only; on Pig on Hive tasks or jobs. These MapReduce sequences can be combined with forks and

path joins.

**C-** Sequences of MapReduce and Pig jobs. These are limited to linear sequences of actions with exception handlers but no forks.

**D-** Iterntive repetition of MapReduce jobs until a desired answer or state is reached.

## Answer:

A

## Explanation:

Oozie workflow is a collection of actions (i.e. Hadoop Map/Reduce jobs, Pig jobs) arranged in a control dependency DAG (Direct Acyclic Graph), specifying a sequence of actions execution. This graph is specified in hPDL (a XML Process Definition Language).

hPDL is a fairly compact language, using a limited amount of flow control and action nodes. Control nodes define the flow of execution and include beginning and end of a workflow (start, end and fail nodes) and mechanisms to control the workflow execution path ( decision, fork and join nodes).

Workflow definitions

Currently running workflow instances, including instance states and variables

Note: Oozie is a Java Web-Application that runs in a Java servlet-container - Tomcat and uses a database to store:

# Question 5

**Question Type:** MultipleChoice

In a large MapReduce job with m mappers and n reducers, how many distinct copy operations will there be in the sort/shuffle phase?

## Options:

**A-** mXn (i.e., m multiplied by n)

**B-** n

**C-** m

**D-** m+n (i.e., m plus n)

**E-** mn (i.e., m to the power of n)

## Answer:

A

## Explanation:

A MapReduce job with m mappers and r reducers involves up to m * r distinct copy operations, since each mapper may have intermediate output going to every reducer.

# Question 6

**Question Type:** MultipleChoice

Analyze each scenario below and indentify which best describes the behavior of the default partitioner?

## Options:

**A-** The default partitioner assigns key-values pairs to reduces based on an internal random number generator.

**B-** The default partitioner implements a round-robin strategy, shuffling the key-value pairs to each reducer in turn. This ensures an event partition of the key space.

**C-** The default partitioner computes the hash of the key. Hash values between specific ranges are associated with different buckets, and each bucket is assigned to a specific reducer.

**D-** The default partitioner computes the hash of the key and divides that valule modulo the number of reducers. The result determines the reducer assigned to process the key-value pair.

**E-** The default partitioner computes the hash of the value and takes the mod of that value with the number of reducers. The result determines the reducer assigned to process the key-value pair.

## Answer:

D

## Explanation:

The default partitioner computes a hash value for the key and assigns the partition based on this result.

The default Partitioner implementation is called HashPartitioner. It uses the hashCode() method of the key objects modulo the number of partitions total to determine which partition to send a given (key, value) pair to.

In Hadoop, the default partitioner is HashPartitioner, which hashes a record's key to determine which partition (and thus which reducer) the record belongs in.The number of partition is then equal to the number of reduce tasks for the job.

# Question 7

**Question Type:** MultipleChoice

Table metadata in Hive is:

## Options:

**A-** Stored as metadata on the NameNode.

**B-** Stored along with the data in HDFS.

**C-** Stored in the Metastore.

**D-** Stored in ZooKeeper.

## Answer:

C

## Explanation:

By default, hive use an embedded Derby database to store metadata information. The metastore is the 'glue' between Hive and HDFS. It tells Hive where your data files live in HDFS, what type of data they contain, what tables they belong to, etc.

The Metastore is an application that runs on an RDBMS and uses an open source ORM layer called DataNucleus, to convert object representations into a relational schema and vice versa. They chose this approach as opposed to storing this information in hdfs as they need the Metastore to be very low latency. The DataNucleus layer allows them to plugin many different RDBMS technologies.

Note:

* By default, Hive stores metadata in an embedded Apache Derby database, and other client/server databases like MySQL can optionally be used.

* features of Hive include:

Metadata storage in an RDBMS, significantly reducing the time to perform semantic checks during query execution.

# Question 8

**Question Type:** MultipleChoice

In the reducer, the MapReduce API provides you with an iterator over Writable values. What does calling the next () method return?

## Options:

**A-** It returns a reference to a different Writable object time.

**B-** It returns a reference to a Writable object from an object pool.

**C-** It returns a reference to the same Writable object each time, but populated with different data.

**D-** It returns a reference to a Writable object. The API leaves unspecified whether this is a reused object or a new object.

**E-** It returns a reference to the same Writable object if the next value is the same as the previous value, or a new Writable object otherwise.

## Answer:

C

### Explanation:

Calling Iterator.next() will always return the SAME EXACT instance of IntWritable, with the contents of that instance replaced with the next value.

## Question 9

**Question Type:** MultipleChoice

What types of algorithms are difficult to express in MapReduce v1 (MRv1)?

### Options:

**A-** Algorithms that require applying the same mathematical function to large numbers of individual binary records.

**B-** Relational operations on large amounts of structured and semi-structured data.

**C-** Algorithms that require global, sharing states.

**D-** Large-scale graph algorithms that require one-step link traversal.

**E-** Text analysis algorithms on large collections of unstructured text (e.g, Web crawls).

## Answer:

C

## Explanation:

See 3) below.

Limitations of Mapreduce -- where not to use Mapreduce

While very powerful and applicable to a wide variety of problems, MapReduce is not the answer to every problem. Here are some problems I found where MapReudce is not suited and some papers that address the limitations of MapReuce.

1. Computation depends on previously computed values

If the computation of a value depends on previously computed values, then MapReduce cannot be used. One good example is the Fibonacci series where each value is summation of the previous two values. i.e., f(k+2) = f(k+1) + f(k). Also, if the data set is small enough to be computed on a single machine, then it is better to do it as a single reduce(map(data)) operation rather than going through the entire map reduce process.

2. Full-text indexing or ad hoc searching

The index generated in the Map step is one dimensional, and the Reduce step must not generate a large amount of data or there will be a serious performance degradation. For example, CouchDB's MapReduce may not be a good fit for full-text indexing or ad hoc searching. This is a problem better suited for a tool such as Lucene.

3. Algorithms depend on shared global state

Solutions to many interesting problems in text processing do not require global synchronization. As a result, they can be expressed naturally in MapReduce, since map and reduce tasks run independently and in isolation. However, there are many examples of algorithms that depend crucially on the existence of shared global state during processing, making them difficult to implement in MapReduce (since the single opportunity for global synchronization in MapReduce is the barrier between the map and reduce phases of processing)