CCD-410 Q&As

Cloudera Certified Developer for Apache Hadoop (CCDH)

Pass Cloudera CCD-410 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.passapply.com/ccd-410.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Cloudera Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

In a MapReduce job, the reducer receives all values associated with same key. Which statement best describes the ordering of these values?

A. The values are in sorted order.

B. The values are arbitrarily ordered, and the ordering may vary from run to run of the same MapReduce job.

C. The values are arbitrary ordered, but multiple runs of the same MapReduce job will always have the same ordering.

D. Since the values come from mapper outputs, the reducers will receive contiguous sections of sorted values.

Correct Answer: B

Note:

*

Input to the Reducer is the sorted output of the mappers.

*

The framework calls the application\\'s Reduce function once for each unique key in the sorted order.

*

Example:

For the given sample input the first map emits:

The second map emits:

QUESTION 2

You have written a Mapper which invokes the following five calls to the OutputColletor.collect method:

output.collect (new Text ("Apple"), new Text ("Red")) ;

output.collect (new Text ("Banana"), new Text ("Yellow")) ; output.collect (new Text ("Apple"), new Text ("Yellow")) ; output.collect (new Text ("Cherry"), new Text ("Red")) ;

output.collect (new Text ("Apple"), new Text ("Green")) ;

How many times will the Reducer\\'s reduce method be invoked?

A. 6

B. 3

C. 1

D. 0

E. 5

Correct Answer: B

reduce() gets called once for each [key, (list of values)] pair. To explain, let\\'s say you called:

out.collect(new Text("Car"),new Text("Subaru");

out.collect(new Text("Car"),new Text("Honda");

out.collect(new Text("Car"),new Text("Ford");

out.collect(new Text("Truck"),new Text("Dodge");

out.collect(new Text("Truck"),new Text("Chevy");

Then reduce() would be called twice with the pairs

reduce(Car,)

reduce(Truck,)

Reference: Mapper output.collect()?

QUESTION 3

You have the following key-value pairs as output from your Map task:

(the, 1) (fox, 1) (faster, 1) (than, 1) (the, 1) (dog, 1)

How many keys will be passed to the Reducer\\'s reduce method?

A. Six

B. Five

C. Four

D. Two

E. One

F. Three

Correct Answer: B

Only one key value pair will be passed from the two (the, 1) key value pairs.

QUESTION 4

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

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).

Correct Answer: C

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)

Reference: Limitations of Mapreduce where not to use Mapreduce

QUESTION 5

The Hadoop framework provides a mechanism for coping with machine issues such as faulty configuration or impending hardware failure. MapReduce detects that one or a number of machines are performing poorly and starts more copies of a map or reduce task. All the tasks run simultaneously and the task finish first are used. This is called:

A. Combine

B. IdentityMapper

C. IdentityReducer

D. Default Partitioner

E. Speculative Execution

Correct Answer: E

Speculative execution: One problem with the Hadoop system is that by dividing the tasks across many nodes, it is possible for a few slow nodes to rate-limit the rest of the program. For example if one node has a slow disk controller, then it may be reading its input at only 10% the speed of all the other nodes. So when 99 map tasks are already complete, the system is still waiting for the final map task to check in, which takes much longer than all the other nodes.

By forcing tasks to run in isolation from one another, individual tasks do not know where their inputs come from. Tasks trust the Hadoop platform to just deliver the appropriate input. Therefore, the same input can be processed multiple times in parallel, to exploit differences in machine capabilities. As most of the tasks in a job are coming to a close, the Hadoop platform will schedule redundant copies of the remaining tasks across several nodes which do not have other work to perform. This process is known as speculative execution. When tasks complete, they announce this fact to the JobTracker. Whichever copy of a task finishes first becomes the definitive copy. If other copies were executing speculatively, Hadoop tells the TaskTrackers to abandon the tasks and discard their outputs. The Reducers then receive their inputs from whichever Mapper completed successfully, first.

Reference: Apache Hadoop, Module 4: MapReduce

Note:

*

Hadoop uses "speculative execution." The same task may be started on multiple boxes. The first one to

finish wins, and the other copies are killed.

Failed tasks are tasks that error out.

*

There are a few reasons Hadoop can kill tasks by his own decisions:

a) Task does not report progress during timeout (default is 10 minutes)

b) FairScheduler or CapacityScheduler needs the slot for some other pool (FairScheduler) or queue

(CapacityScheduler).

c) Speculative execution causes results of task not to be needed since it has completed on other place.

Reference: Difference failed tasks vs killed tasks