CCD-410$^{Q\&As}$

Cloudera Certified Developer for Apache Hadoop (CCDH)

Pass Cloudera CCD-410 Exam with 100% Guarantee

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 want to count the number of occurrences for each unique word in the supplied input data. You\\'ve decided to implement this by having your mapper tokenize each word and emit a literal value 1, and then have your reducer increment a counter for each literal 1 it receives. After successful implementing this, it occurs to you that you could optimize this by specifying a combiner. Will you be able to reuse your existing Reduces as your combiner in this case and why or why not?

A. Yes, because the sum operation is both associative and commutative and the input and output types to the reduce method match.

B. No, because the sum operation in the reducer is incompatible with the operation of a Combiner.

C. No, because the Reducer and Combiner are separate interfaces.

D. No, because the Combiner is incompatible with a mapper which doesn\\'t use the same data type for both the key and value.

E. Yes, because Java is a polymorphic object-oriented language and thus reducer code can be reused as a combiner.

Correct Answer: A

Combiners are used to increase the efficiency of a MapReduce program. They are used to aggregate intermediate map output locally on individual mapper outputs. Combiners can help you reduce the amount of data that needs to be transferred across to the reducers. You can use your reducer code as a combiner if the operation performed is commutative and associative. The execution of combiner is not guaranteed, Hadoop may or may not execute a combiner. Also, if required it may execute it more then 1 times. Therefore your MapReduce jobs should not depend on the combiners execution.

Reference: 24 Interview Questions and Answers for Hadoop MapReduce developers, What are combiners? When should I use a combiner in my MapReduce Job?

QUESTION 3

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 4

You\\'ve written a MapReduce job that will process 500 million input records and generated 500 million keyvalue pairs. The data is not uniformly distributed. Your MapReduce job will create a significant amount of intermediate data that it needs to transfer between mappers and reduces which is a potential bottleneck. A custom implementation of which interface is most likely to reduce the amount of intermediate data transferred across the network?

A. Partitioner

B. OutputFormat

C. WritableComparable

D. Writable

E. InputFormat

F. Combiner

Correct Answer: F

Combiners are used to increase the efficiency of a MapReduce program. They are used to aggregate intermediate map output locally on individual mapper outputs. Combiners can help you reduce the amount of data that needs to be transferred across to the reducers. You can use your reducer code as a combiner if the operation performed is commutative and associative.

Reference: 24 Interview Questions and Answers for Hadoop MapReduce developers, What are combiners? When should I use a combiner in my MapReduce Job?

QUESTION 5

You have just executed a MapReduce job. Where is intermediate data written to after being emitted from the Mapper\\'s map method?

A. Intermediate data in streamed across the network from Mapper to the Reduce and is never written to disk.

B. Into in-memory buffers on the TaskTracker node running the Mapper that spill over and are written into HDFS.

C. Into in-memory buffers that spill over to the local file system of the TaskTracker node running the Mapper.

D. Into in-memory buffers that spill over to the local file system (outside HDFS) of the TaskTracker node running the Reducer

E. Into in-memory buffers on the TaskTracker node running the Reducer that spill over and are written into HDFS.

Correct Answer: C

The mapper output (intermediate data) is stored on the Local file system (NOT HDFS) of each individual mapper nodes. This is typically a temporary directory location which can be setup in config by the hadoop administrator. The intermediate data is cleaned up after the Hadoop Job completes.

Reference: 24 Interview Questions and Answers for Hadoop MapReduce developers, Where is the Mapper Output (intermediate kay-value data) stored ?