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.geekcert.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

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 2

Which process describes the lifecycle of a Mapper?

A. The JobTracker calls the TaskTracker\\'s configure () method, then its map () method and finally its close () method.

B. The TaskTracker spawns a new Mapper to process all records in a single input split.

C. The TaskTracker spawns a new Mapper to process each key-value pair.

D. The JobTracker spawns a new Mapper to process all records in a single file.

Correct Answer: B

For each map instance that runs, the TaskTracker creates a new instance of your mapper. Note:

*

The Mapper is responsible for processing Key/Value pairs obtained from the InputFormat. The mapper may perform a number of Extraction and Transformation functions on the Key/Value pair before ultimately outputting none, one or many Key/Value pairs of the same, or different Key/Value type.

*

With the new Hadoop API, mappers extend the org.apache.hadoop.mapreduce.Mapper class. This class defines an \\'Identity\\' map function by default - every input Key/Value pair obtained from the InputFormat is written out.

Examining the run() method, we can see the lifecycle of the mapper:

/**

*

Expert users can override this method for more complete control over the

*

execution of the Mapper.

*

@param context

*

@throws IOException

*/

public void run(Context context) throws IOException, InterruptedException {

setup(context);

while (context.nextKeyValue()) {

map(context.getCurrentKey(), context.getCurrentValue(), context);

}

cleanup(context);

}

setup(Context) - Perform any setup for the mapper. The default implementation is a no-op method.

map(Key, Value, Context) - Perform a map operation in the given Key / Value pair. The default

implementation calls Context.write(Key, Value)

cleanup(Context) - Perform any cleanup for the mapper. The default implementation is a no-op method.

Reference: Hadoop/MapReduce/Mapper

QUESTION 3

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 4

Your cluster\\'s HDFS block size in 64MB. You have directory containing 100 plain text files, each of which is 100MB in size. The InputFormat for your job is TextInputFormat. Determine how many Mappers will run?

A. 64

B. 100

C. 200

D. 640

Correct Answer: C

Each file would be split into two as the block size (64 MB) is less than the file size (100 MB), so 200 mappers would be running.

Note:

If you\\'re not compressing the files then hadoop will process your large files (say 10G), with a number of mappers related to the block size of the file.

Say your block size is 64M, then you will have ~160 mappers processing this 10G file (160*64 ~= 10G). Depending on how CPU intensive your mapper logic is, this might be an acceptable blocks size, but if you find that your mappers are executing in sub minute times, then you might want to increase the work done by each mapper (by increasing the block size to 128, 256, 512m - the actual size depends on how you intend to process the data).

Reference: http://stackoverflow.com/questions/11014493/hadoop-mapreduce-appropriate-input-files-size (first answer, second paragraph)

QUESTION 5

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()?