Hortonworks

Exam HDPCD

Hortonworks Data Platform Certified Developer

Version: 5.0

[Total Questions: 108]

Question No : 1

Workflows expressed in Oozie can contain:

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

Reference: Introduction to Oozie

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

Question No : 2

You are developing a combiner that takes as input Text keys, IntWritable values, and emits Text keys, IntWritable values. Which interface should your class implement?

A. Combiner <Text, IntWritable, Text, IntWritable>
B. Mapper <Text, IntWritable, Text, IntWritable>

C. Reducer <Text, Text, IntWritable, IntWritable>
D. Reducer <Text, IntWritable, Text, IntWritable>
E. Combiner <Text, Text, IntWritable, IntWritable>

Answer: D

Question No : 3

Which TWO of the following statements are true regarding Hive? Choose 2 answers

A. Useful for data analysts familiar with SQL who need to do ad-hoc queries
B. Offers real-time queries and row level updates
C. Allows you to define a structure for your unstructured Big Data
D. Is a relational database

Answer: A,C

Question No : 4

You need to create a job that does frequency analysis on input data. You will do this by writing a Mapper that uses TextInputFormat and splits each value (a line of text from an input file) into individual characters. For each one of these characters, you will emit the character as a key and an InputWritable as the value. As this will produce proportionally more intermediate data than input data, which two resources should you expect to be bottlenecks?

A. Processor and network I/O
B. Disk I/O and network I/O
C. Processor and RAM
D. Processor and disk I/O

Answer: B

Question No : 5

Which one of the following classes would a Pig command use to store data in a table defined in HCatalog?

A. org.apache.hcatalog.pig.HCatOutputFormat
B. org.apache.hcatalog.pig.HCatStorer
C. No special class is needed for a Pig script to store data in an HCatalog table
D. Pig scripts cannot use an HCatalog table

Answer: B

Question No : 6

All keys used for intermediate output from mappers must:

A. Implement a splittable compression algorithm.
B. Be a subclass of FileInputFormat.
C. Implement WritableComparable.
D. Override isSplitable.
E. Implement a comparator for speedy sorting.

Answer: C

Explanation: The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and value classes have to be serializable by the framework and hence need to implement the Writable interface. Additionally, the key classes have to implement the WritableComparable interface to facilitate sorting by the framework.

Reference: MapReduce Tutorial

Question No : 7

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).

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)

Reference: Limitations of Mapreduce – where not to use Mapreduce

Question No : 8

How are keys and values presented and passed to the reducers during a standard sort and

shuffle phase of MapReduce?

A. Keys are presented to reducer in sorted order; values for a given key are not sorted.
B. Keys are presented to reducer in sorted order; values for a given key are sorted in ascending order.
C. Keys are presented to a reducer in random order; values for a given key are not sorted.
D. Keys are presented to a reducer in random order; values for a given key are sorted in ascending order.

Answer: A

Explanation: Reducer has 3 primary phases:

1. Shuffle

The Reducer copies the sorted output from each Mapper using HTTP across the network.

2. Sort

The framework merge sorts Reducer inputs by keys (since different Mappers may have output the same key).

The shuffle and sort phases occur simultaneously i.e. while outputs are being fetched they are merged.

SecondarySort

To achieve a secondary sort on the values returned by the value iterator, the application should extend the key with the secondary key and define a grouping comparator. The keys will be sorted using the entire key, but will be grouped using the grouping comparator to decide which keys and values are sent in the same call to reduce.

3. Reduce

In this phase the reduce(Object, Iterable, Context) method is called for each <key, (collection of values)> in the sorted inputs.

The output of the reduce task is typically written to a RecordWriter via TaskInputOutputContext.write(Object, Object).

The output of the Reducer is not re-sorted.

Reference: org.apache.hadoop.mapreduce, Class Reducer<KEYIN,VALUEIN,KEYOUT,VALUEOUT>

Question No : 9

Which best describes how TextInputFormat processes input files and line breaks?

A. Input file splits may cross line breaks. A line that crosses file splits is read by the RecordReader of the split that contains the beginning of the broken line.
B. Input file splits may cross line breaks. A line that crosses file splits is read by the RecordReaders of both splits containing the broken line.
C. The input file is split exactly at the line breaks, so each RecordReader will read a series of complete lines.
D. Input file splits may cross line breaks. A line that crosses file splits is ignored.
E. Input file splits may cross line breaks. A line that crosses file splits is read by the RecordReader of the split that contains the end of the broken line.

Answer: A
Reference: How Map and Reduce operations are actually carried out

Question No : 10

To process input key-value pairs, your mapper needs to lead a 512 MB data file in memory. What is the best way to accomplish this?

A. Serialize the data file, insert in it the JobConf object, and read the data into memory in the configure method of the mapper.
B. Place the data file in the DistributedCache and read the data into memory in the map method of the mapper.
C. Place the data file in the DataCache and read the data into memory in the configure method of the mapper.
D. Place the data file in the DistributedCache and read the data into memory in the configure method of the mapper.

Answer: C

Question No : 11

Consider the following two relations, A and B.

```
A = LOAD 'data1' AS (a1:int,a2:chararray);
DUMP A;
(1,apple)
(3,orange)
(4,peach)
(2,cherry)

B = LOAD 'data2' AS (b1:chararray,b2:int);
DUMP B;
(Jim,2)
(Brian,4)
(Kim,0)
(Terry,3)
(Chris,2)
```

Which Pig statement combines A by its first field and B by its second field?

A. C = DOIN B BY a1, A by b2;
B. C = JOIN A by al, B by b2;
C. C = JOIN A a1, B b2;
D. C = JOIN A SO, B $1;

Answer: B

Question No : 12

What is the disadvantage of using multiple reducers with the default HashPartitioner and distributing your workload across you cluster?

A. You will not be able to compress the intermediate data.
B. You will longer be able to take advantage of a Combiner.
C. By using multiple reducers with the default HashPartitioner, output files may not be in globally sorted order.
D. There are no concerns with this approach. It is always advisable to use multiple reduces.

Answer: C

Explanation: Multiple reducers and total ordering

If your sort job runs with multiple reducers (either because mapreduce.job.reduces in mapred-site.xml has been set to a number larger than 1, or because you've used the -r option to specify the number of reducers on the command-line), then by default Hadoop will use the HashPartitioner to distribute records across the reducers. Use of the HashPartitioner means that you can't concatenate your output files to create a single sorted output file. To do this you'll need total ordering,

Reference: Sorting text files with MapReduce

Question No : 13

Identify the MapReduce v2 (MRv2 / YARN) daemon responsible for launching application containers and monitoring application resource usage?

A. ResourceManager
B. NodeManager
C. ApplicationMaster
D. ApplicationMasterService
E. TaskTracker
F. JobTracker

Answer: B

Reference: Apache Hadoop YARN – Concepts & Applications

Question No : 14

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

Answer: B

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

Question No : 15

What data does a Reducer reduce method process?

A. All the data in a single input file.
B. All data produced by a single mapper.
C. All data for a given key, regardless of which mapper(s) produced it.
D. All data for a given value, regardless of which mapper(s) produced it.

Answer: C

Explanation: Reducing lets you aggregate values together. A reducer function receives an iterator of input values from an input list. It then combines these values together, returning a single output value.

All values with the same key are presented to a single reduce task.

Reference: Yahoo! Hadoop Tutorial, Module 4: MapReduce