Free Questions for HDPCD by certsdeals

Shared by William on 12-12-2023

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

MapReduce v2 (MRv2/YARN) splits which major functions of the JobTracker into separate daemons? Select two.

Options:

A- Heath states checks (heartbeats)

B- Resource management

C- Job scheduling/monitoring

D- Job coordination between the ResourceManager and NodeManager

E- Launching tasks

F- Managing file system metadata

G- MapReduce metric reporting

H- Managing tasks

Answer:

B, C

Explanation:

The fundamental idea of MRv2 is to split up the two major functionalities of the JobTracker, resource management and job scheduling/monitoring, into separate daemons. The idea is to have a global ResourceManager (RM) and per-application ApplicationMaster (AM). An application is either a single job in the classical sense of Map-Reduce jobs or a DAG of jobs.

Note:

The central goal of YARN is to clearly separate two things that are unfortunately smushed together in current Hadoop, specifically in (mainly) JobTracker:

/ Monitoring the status of the cluster with respect to which nodes have which resources available. Under YARN, this will be global.

/ Managing the parallelization execution of any specific job. Under YARN, this will be done separately for each job.

Question 2

Question Type: **MultipleChoice**

In a MapReduce job with 500 map tasks, how many map task attempts will there be?

Options:

A- It depends on the number of reduces in the job.

B- Between 500 and 1000.

C- At most 500.

D- At least 500.

E- Exactly 500.

Answer:

D

Explanation:

From Cloudera Training Course:

Task attempt is a particular instance of an attempt to execute a task

-- There will be at least as many task attempts as there are tasks

-- If a task attempt fails, another will be started by the JobTracker

-- Speculative execution can also result in more task attempts than completed tasks

Question 3

Question Type: MultipleChoice

A combiner reduces:

Options:

A- The number of values across different keys in the iterator supplied to a single reduce method call.

B- The amount of intermediate data that must be transferred between the mapper and reducer.

C- The number of input files a mapper must process.

D- The number of output files a reducer must produce.

Answer:

B

Explanation:

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.

Question 4

Question Type: MultipleChoice

You write MapReduce job to process 100 files in HDFS. Your MapReduce algorithm uses TextInputFormat: the mapper applies a regular expression over input values and emits key-values pairs with the key consisting of the matching text, and the value containing the filename and byte offset. Determine the difference between setting the number of reduces to one and settings the number of reducers to zero.

Options:

A- There is no difference in output between the two settings.

B- With zero reducers, no reducer runs and the job throws an exception. With one reducer, instances of matching patterns are stored in a single file on HDFS.

C- With zero reducers, all instances of matching patterns are gathered together in one file on HDFS. With one reducer, instances of matching patterns are stored in multiple files on HDFS.

D- With zero reducers, instances of matching patterns are stored in multiple files on HDFS. With one reducer, all instances of matching

patterns are gathered together in one file on HDFS.

Answer:

D

Explanation:

* It is legal to set the number of reduce-tasks to zero if no reduction is desired.

In this case the outputs of the map-tasks go directly to the FileSystem, into the output path set by setOutputPath(Path). The framework does not sort the map-outputs before writing them out to the FileSystem.

* Often, you may want to process input data using a map function only. To do this, simply set mapreduce.job.reduces to zero. The MapReduce framework will not create any reducer tasks. Rather, the outputs of the mapper tasks will be the final output of the job.

Note:

Reduce

In this phase the reduce(WritableComparable, Iterator, OutputCollector, Reporter) method is called for each <key, (list of values)> pair in the grouped inputs.

The output of the reduce task is typically written to the FileSystem via OutputCollector.collect(WritableComparable, Writable).

Applications can use the Reporter to report progress, set application-level status messages and update Counters, or just indicate that they are alive.

The output of the Reducer is not sorted.

Question 5

Question Type: MultipleChoice

You have a directory named jobdata in HDFS that contains four files: _first.txt, second.txt, .third.txt and #data.txt. How many files will be processed by the FileInputFormat.setInputPaths () command when it's given a path object representing this directory?

Options:

A- Four, all files will be processed

B- Three, the pound sign is an invalid character for HDFS file names

C- Two, file names with a leading period or underscore are ignored

D- None, the directory cannot be named jobdata

E- One, no special characters can prefix the name of an input file

Answer:

C

Explanation:

Files starting with '_' are considered 'hidden' like unix files starting with '.'.

characters are allowed in HDFS file names.

Question 6

Question Type: MultipleChoice

Identify the tool best suited to import a portion of a relational database every day as files into HDFS, and generate Java classes to interact with that imported data?

Options:

A- Oozie

B- Flume

C- Pig

D- Hue

E- Hive

F- Sqoop

G- fuse-dfs

Answer:

F

Explanation:

Sqoop (''SQL-to-Hadoop'') is a straightforward command-line tool with the following capabilities:

Imports individual tables or entire databases to files in HDFS

Generates Java classes to allow you to interact with your imported data

Provides the ability to import from SQL databases straight into your Hive data warehouse

Note:

Data Movement Between Hadoop and Relational Databases

Data can be moved between Hadoop and a relational database as a bulk data transfer, or relational tables can be accessed from within a MapReduce map function.

Note:

* Cloudera's Distribution for Hadoop provides a bulk data transfer tool (i.e., Sqoop) that imports individual tables or entire databases into HDFS files. The tool also generates Java classes that support interaction with the imported data. Sqoop supports all relational databases over JDBC, and Quest Software provides a connector (i.e., OraOop) that has been optimized for access to data residing in Oracle databases.

Question 7

Question Type: **MultipleChoice**

You use the hadoop fs --put command to write a 300 MB file using and HDFS block size of 64 MB. Just after this command has finished writing 200 MB of this file, what would another user see when trying to access this life?

Options:

A- They would see Hadoop throw an ConcurrentFileAccessException when they try to access this file.

B- They would see the current state of the file, up to the last bit written by the command.

C- They would see the current of the file through the last completed block.

D- They would see no content until the whole file written and closed.

Answer:

C

Question 8

Question Type: **MultipleChoice**

Which project gives you a distributed, Scalable, data store that allows you random, realtime read/write access to hundreds of terabytes of data?

Options:

A- HBase

B- Hue

C- Pig

D- Hive

E- Oozie

F- Flume

G- Sqoop

Answer:

A

Explanation:

Use Apache HBase when you need random, realtime read/write access to your Big Data.

Note: This project's goal is the hosting of very large tables -- billions of rows X millions of columns -- atop clusters of commodity hardware. Apache HBase is an open-source, distributed, versioned, column-oriented store modeled after Google's Bigtable: A Distributed Storage System for Structured Data by Chang et al. Just as Bigtable leverages the distributed data storage provided by the Google File System, Apache HBase provides Bigtable-like capabilities on top of Hadoop and HDFS.

Features

Linear and modular scalability.

Strictly consistent reads and writes.

Automatic and configurable sharding of tables

Automatic failover support between RegionServers.

Convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables.

Easy to use Java API for client access.

Block cache and Bloom Filters for real-time queries.

Query predicate push down via server side Filters

Thrift gateway and a REST-ful Web service that supports XML, Protobuf, and binary data encoding options

Extensible jruby-based (JIRB) shell

Support for exporting metrics via the Hadoop metrics subsystem to files or Ganglia; or via JMX