Databricks

DATABRICKS-CERTIFIED-ASSOCIATE-DEVELOPER-FOR-APACHE-SPARK-3.0 Exam

Databricks Certified Associate Developer for Apache Spark 3.0

Questions & Answers
Demo

Version: 4.0

Question: 1

Which of the following options describes the responsibility of the executors in Spark?

A. The executors accept jobs from the driver, analyze those jobs, and return results to the driver.
B. The executors accept tasks from the driver, execute those tasks, and return results to the cluster manager.
C. The executors accept tasks from the driver, execute those tasks, and return results to the driver.
D. The executors accept tasks from the cluster manager, execute those tasks, and return results to the driver.
E. The executors accept jobs from the driver, plan those jobs, and return results to the cluster manager.

Answer: C

Explanation:
More info: Running Spark: an overview of Spark's runtime architecture - Manning (https://bit.ly/2RPmJn9)

Question: 2

Which of the following describes the role of tasks in the Spark execution hierarchy?

A. Tasks are the smallest element in the execution hierarchy.
B. Within one task, the slots are the unit of work done for each partition of the data.
C. Tasks are the second-smallest element in the execution hierarchy.
D. Stages with narrow dependencies can be grouped into one task.
E. Tasks with wide dependencies can be grouped into one stage.

Answer: A

Explanation:
Stages with narrow dependencies can be grouped into one task.
Wrong, tasks with narrow dependencies can be grouped into one stage.
Tasks with wide dependencies can be grouped into one stage.
Wrong, since a wide transformation causes a shuffle which always marks the boundary of a stage. So,

you cannot bundle multiple tasks that have wide dependencies into a stage.
Tasks are the second-smallest element in the execution hierarchy.
No, they are the smallest element in the execution hierarchy.
Within one task, the slots are the unit of work done for each partition of the data.
No, tasks are the unit of work done per partition. Slots help Spark parallelize work. An executor can have multiple slots which enable it to process multiple tasks in parallel.

Question: 3

Which of the following describes the role of the cluster manager?

A. The cluster manager schedules tasks on the cluster in client mode.
B. The cluster manager schedules tasks on the cluster in local mode.
C. The cluster manager allocates resources to Spark applications and maintains the executor processes in client mode.
D. The cluster manager allocates resources to Spark applications and maintains the executor processes in remote mode.
E. The cluster manager allocates resources to the DataFrame manager.

Answer: C

Explanation:
The cluster manager allocates resources to Spark applications and maintains the executor processes in client mode.
Correct. In cluster mode, the cluster manager is located on a node other than the client machine. From there it starts and ends executor processes on the cluster nodes as required by the Spark application running on the Spark driver.
The cluster manager allocates resources to Spark applications and maintains the executor processes in remote mode.
Wrong, there is no "remote" execution mode in Spark. Available execution modes are local, client, and cluster.
The cluster manager allocates resources to the DataFrame manager
Wrong, there is no "DataFrame manager" in Spark.
The cluster manager schedules tasks on the cluster in client mode.
No, in client mode, the Spark driver schedules tasks on the cluster – not the cluster manager.
The cluster manager schedules tasks on the cluster in local mode.
Wrong: In local mode, there is no "cluster". The Spark application is running on a single machine, not on a cluster of machines.

Question: 4

Which of the following is the idea behind dynamic partition pruning in Spark?

A. Dynamic partition pruning is intended to skip over the data you do not need in the results of a query.

B. Dynamic partition pruning concatenates columns of similar data types to optimize join performance.
C. Dynamic partition pruning performs wide transformations on disk instead of in memory.
D. Dynamic partition pruning reoptimizes physical plans based on data types and broadcast variables.
E. Dynamic partition pruning reoptimizes query plans based on runtime statistics collected during query execution.

Answer: A

Explanation:
Dynamic partition pruning reoptimizes query plans based on runtime statistics collected during query execution.
No – this is what adaptive query execution does, but not dynamic partition pruning.
Dynamic partition pruning concatenates columns of similar data types to optimize join performance.
Wrong, this answer does not make sense, especially related to dynamic partition pruning.
Dynamic partition pruning reoptimizes physical plans based on data types and broadcast variables.
It is true that dynamic partition pruning works in joins using broadcast variables. This actually happens in both the logical optimization and the physical planning stage. However, data types do not play a role for the reoptimization.
Dynamic partition pruning performs wide transformations on disk instead of in memory.
This answer does not make sense. Dynamic partition pruning is meant to accelerate Spark – performing any transformation involving disk instead of memory resources would decelerate Spark and certainly achieve the opposite effect of what dynamic partition pruning is intended for.

Question: 5

Which of the following is one of the big performance advantages that Spark has over Hadoop?

A. Spark achieves great performance by storing data in the DAG format, whereas Hadoop can only use parquet files.
B. Spark achieves higher resiliency for queries since, different from Hadoop, it can be deployed on Kubernetes.
C. Spark achieves great performance by storing data and performing computation in memory, whereas large jobs in Hadoop require a large amount of relatively slow disk I/O operations.
D. Spark achieves great performance by storing data in the HDFS format, whereas Hadoop can only use parquet files.
E. Spark achieves performance gains for developers by extending Hadoop's DataFrames with a user-friendly API.

Answer: C

Explanation:
Spark achieves great performance by storing data in the DAG format, whereas Hadoop can only use parquet files.
Wrong, there is no "DAG format". DAG stands for "directed acyclic graph". The DAG is a means of

representing computational steps in Spark. However, it is true that Hadoop does not use a DAG.

The introduction of the DAG in Spark was a result of the limitation of Hadoop's map reduce framework in which data had to be written to and read from disk continuously.

Graph DAG in Apache Spark - DataFlair
Spark achieves great performance by storing data in the HDFS format, whereas Hadoop can only use parquet files.
No. Spark can certainly store data in HDFS (as well as other formats), but this is not a key performance advantage over Hadoop. Hadoop can use multiple file formats, not only parquet.
Spark achieves higher resiliency for queries since, different from Hadoop, it can be deployed on Kubernetes.
No, resiliency is not asked for in the question. The Question: is about
performance improvements. Both Hadoop and Spark can be deployed on Kubernetes.
Spark achieves performance gains for developers by extending Hadoop's DataFrames with a user-friendly API.
No. DataFrames are a concept in Spark, but not in Hadoop.