DUMPSBOSS.

HCIP-Big Data Developer V2.0

Huawei H13-723_V2.0

Version Demo

Total Demo Questions: 39

Total Premium Questions: 391

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Big Data Development Overview	53
Topic 2, Hadoop Development	111
Topic 3, Spark Development	49
Topic 4, Flink Development	11
Topic 5, HBase Development	47
Topic 6, Hive Development	44
Topic 7, Kafka Development	22
Topic 8, Elasticsearch Development	22
Topic 9, Mix Questions	32
Total	391

QUESTION NO: 1

RDDavailable fromHadoopCompatible file system generation, after generation, it can be generated by callingRDDcalculation child pairRDDThe data is partially updated.

A. True

B. False

ANSWER: B

Explanation:

False is correct because an RDD is an immutable distributed collection. An RDD can be created by reading data from a Hadoop-compatible file system, such as HDFS, and Spark can then apply transformations to derive further RDDs. However, neither the source RDD nor a derived pair RDD supports an in-place update of only part of its data. Operations that appear to modify data actually define a new RDD containing the transformed records, while the earlier RDD remains unchanged. This immutability is a core part of Spark's fault-tolerance model: Spark records the lineage of transformations and can recompute lost partitions from their parent RDDs rather than relying on mutable intermediate state. For key-value data, transformations such as `mapValues`, `reduceByKey`, and `join` produce new pair RDDs; they do not partially update an existing pair RDD. If an application requires mutable state or updates, it must use an appropriate external storage system or create a new dataset representing the updated state. See the Apache Spark RDD programming guide at https://spark.apache.org/docs/latest/rdd-programming-guide.html and the Spark RDD API documentation at https://spark.apache.org/docs/latest/api/java/org/apache/spark/rdd/RDD.html.

QUESTION NO: 2

Regarding the Redis cluster topology information, which of the following descriptions is correct?

A. The client caches the topology information of the cluster.

B. The server caches the topology information of the cluster

C. both are

D. neither

ANSWER: A B C

Explanation:

Redis Cluster topology is maintained at both ends of the connection. Cluster nodes maintain their own view of the cluster configuration, including node identities, addresses, roles, and hash-slot assignments. This information is persisted in each node's cluster configuration file (commonly `nodes.conf`) and is exchanged among nodes through the cluster bus, allowing the topology to converge and remain available to serving nodes.

Clients also maintain a cached hash-slot-to-node mapping. A cluster-aware client uses that mapping to send a key directly to the node responsible for its hash slot rather than querying an arbitrary node for every command. When a slot is moved or the cached mapping is stale, Redis returns redirection information such as `MOVED`; the client updates its topology cache and retries against the indicated node. Therefore, the statements that the client caches topology information, the server caches topology information, and both do are correct descriptions of normal Redis Cluster behavior. See the Redis Cluster specification and the Redis Cluster management documentation.

QUESTION NO: 3

existMapReduceIn application development of , which of the followingJAVA, Class responsible for managing and running a computational task?

A. Job

B. Context

C. FileSystem

D. Configuration

ANSWER: A

Explanation:

`Job` is the Hadoop MapReduce Java class used by a client application to define, configure, submit, monitor, and control a MapReduce computation. A developer typically creates a `Job` instance from a Hadoop `Configuration`, then associates the application's mapper, reducer, input format, output format, key/value classes, input paths, and output path with that instance. Once these settings are complete, the application calls methods such as `waitForCompletion` to submit the job and wait while Hadoop executes it across the cluster. The `Job` object also provides status and progress information and supports operations such as checking whether execution has completed or terminating the submitted job. Thus, it is the application-facing abstraction that manages the lifecycle of the MapReduce computational job, from setup through execution monitoring. Hadoop's MapReduce tutorial demonstrates creating a `Job`, setting processing classes and paths, and invoking `waitForCompletion`. The official API further describes `Job` as the class for submitting jobs and tracking their progress. See the Apache Hadoop MapReduce Tutorial and the Job API documentation.

QUESTION NO: 4

In HBase, which operation is used to retrieve a single row by its RowKey?

A. Put

B. Scan

C. Get

D. Delete

ANSWER: C

Explanation:

Get is correct. An HBase `Get` request reads data for one specified row key. The request can further restrict the returned result to selected column families, qualifiers, timestamps, or versions. This is the normal API for low-latency random reads when the target RowKey is known.

A `Scan` is used to iterate through a range of rows or a larger table section. `Put` writes data, while `Delete` adds deletion markers for specified HBase data. See the HBase Get API documentation.

QUESTION NO: 5

FusionInsight ManagerRegarding the management operations of services, which of the following statements is wrong?

A. Can start, stop and restart the service

B. Services can be added and uninstalled.

C. Uncommon services can be set to hide or show

D. Can view the current status of the service

ANSWER: C

Explanation:

Uncommon services can be set to hide or show is the incorrect statement. In FusionInsight Manager, service management is intended to administer the deployed cluster services and their components. Its operational model centers on maintaining the service lifecycle and observing service health: administrators use the Service Management area to access a service, inspect its running and health information, and perform authorized maintenance actions. The product does not define “uncommon service” as a service-management category for which an administrator can toggle visibility. A deployed service remains part of the managed cluster inventory so that its health, alarms, dependencies, configuration, and maintenance state can be consistently tracked by the platform. Interface display preferences or dashboard presentation should not be confused with a supported service-level hide/show operation. Consequently, claiming that uncommon services may be hidden or shown describes a capability outside the documented service-management operations and is the statement that is wrong. Huawei’s FusionInsight documentation portal provides the Administrator Guide and the Service Management procedures for the applicable product version; see the Huawei FusionInsight support portal. Related managed-service administration concepts are also documented in Huawei Cloud MRS guidance at Managing clusters and services.

QUESTION NO: 6

HuaweiFusionInsight HDin the cluster,SparkFrom which of the following services can a service read data? (multiple choice)

A. YARN

B. HDFS

C. Hive

D. HBase

ANSWER: B C D

Explanation:

Spark in a FusionInsight HD cluster can use HDFS, Hive, and HBase as data sources. HDFS provides the distributed file-system storage from which Spark can load files and persisted datasets, including common structured and unstructured formats. Hive is available to Spark through Spark SQL and Hive integration: Spark can use the Hive metastore to resolve table definitions and query the data represented by those tables. HBase can be accessed by Spark through the HBase client and the relevant Spark-HBase integration connector, allowing Spark jobs to scan or retrieve records from HBase tables for distributed processing. These sources reflect the distinct storage and metadata/database services with which Spark commonly integrates in the FusionInsight ecosystem. Huawei’s MRS/FusionInsight documentation provides service documentation for its HDFS, Hive, HBase, and Spark components at Huawei Cloud MRS User Guide. Spark’s Hive support is also documented by the Apache Spark project at Spark SQL Hive Tables documentation.

QUESTION NO: 7

FusionInsight HDsystematicV100R002C60version,HiveOnly supports based onMapReduceEngine query service,

not supported based onSparkEngine query service.

A. True

B. False

ANSWER: B

Explanation:

False is correct. In FusionInsight HD V100R002C60, Hive is not restricted to MapReduce as its query execution engine. The platform supports running Hive queries with Spark as an execution engine, enabling HiveQL workloads to use Spark’s distributed processing capabilities. In practical terms, an administrator or user can configure Hive’s execution-engine setting for Spark and submit compatible Hive SQL statements; Hive then translates the query plan for execution by Spark rather than relying solely on MapReduce. This capability is important for interactive and iterative SQL workloads because Spark can avoid some of the high startup overhead traditionally associated with MapReduce jobs while retaining Hive metadata and SQL compatibility. Huawei’s MapReduce Service documentation describes Hive and Spark as supported big-data

components and provides operational guidance for their use within the same cluster environment. The upstream Hive documentation also describes Hive-on-Spark and the `hive.execution.engine=spark` configuration used to select Spark execution. Therefore, the statement's claim that Spark-engine query service is unsupported is not valid for this FusionInsight HD release. See Huawei Cloud MRS Product Description and Apache Hive: Hive on Spark.

QUESTION NO: 8

About Kafka consumer groups, which of the following statements are correct? (multiple choice)

A. Consumers with the same group ID coordinate partition assignment.

B. One topic partition can be assigned to multiple consumers in the same consumer group at the same time.

C. Different consumer groups can independently consume the same topic.

D. A consumer group with more consumers than partitions can have idle consumers.

ANSWER: A C D

Explanation:

Consumers that use the same group ID form a consumer group. Within one group, a topic partition is assigned to no more than one consumer at a time, so records in that partition are processed by one member of that group. This prevents duplicate processing by multiple consumers in the same group and allows partitions to be processed in parallel when sufficient consumers are available.

Different consumer groups are independent. Each group maintains its own committed offsets and can consume the same topic records for a different business purpose. If a group has more consumers than partitions subscribed to, some consumers remain idle because no partition can be assigned to them. See the Apache Kafka consumer configuration documentation and the Kafka consumer concepts documentation.

QUESTION NO: 9

There are the following business scenarios: User online log files have been stored inHDFSabove, the log file content format the formula is: each online record has three fields, namely name, gender, and online time, and the fields are separated by ",";

It is required to print out all female netizens who spend more than two hours online. Which of the following code snippets can achieve

The above business scenario? (multiple choice)

A. sc.textFile("/data/file/path").map(_.split(",")).map(p => FemaleInfo(p(0), p(1),
p(2).trim.toInt)).toDF.registerTempTable("FemaleInfoTable")
sqlContext.sql("select name, sum(stayTime) as stayTime from FemaleInfoTable where gender='female' group by name").filter("stayTime >= 120").collect.foreach(println)

B. sc.textFile("/data/file/path").map(_.split(",")).map(p=>FemaleInfo
(p(O),p(1),p(2).trim.toInt)).toDF.registerTemp Table("Female infoTable")
sqlContext.sq(l "select name, sum(stay Time)as stay Time from FemaleInfoTable where gender=female).filter("stay Time >=120").collect).foreach(printin)

C. val text = sc.textFile("/data/file/path")
val data = text.filter(_.contains("female"))
val femaleData: RDD[(String, Int)] = data.map { line => val t = line.split(','); (t(0), t(2).toInt) }.reduceByKey(_ + _)
val result = femaleData.filter(line => line._2 > 120)
result.collect.map(x => x._1 + ',' + x._2).foreach(println)

D. val text=sc.textFile("/data/fle/path")
val data=text.filter(_.contains("female"))
Val fermaleData:RDD(String,Int) =data.map(line=>val t=line.split(',')

(t(0),t(2).toInt)
)}
Val result=femaleData.filter(line=>line.2>120)
Rusult.collect.map(x=>x._1 +','+ x._2).foreach(println)

ANSWER: A C

Explanation:

The SQL/DataFrame approach that creates a temporary table, filters records where `gender = 'female'`, aggregates online duration with `sum(stayTime)`, groups by `name`, and retains totals of at least 120 minutes correctly implements the required processing flow. It first converts the comma-separated HDFS input into structured fields, then performs the needed gender filter and per-user aggregation before printing the resulting users. Spark SQL supports this style of temporary-view registration and SQL aggregation; see the Apache Spark SQL getting-started guide.

The RDD approach that filters female records, splits each line into fields, maps each record to a `(name, onlineTime)` pair, and uses `reduceByKey(_ + _)` is also correct. Reducing by key combines every online-duration value belonging to the same user, so the subsequent duration filter is applied to each user's accumulated time rather than to an individual log entry. The final mapping and printing output the user name and total online duration. This is consistent with Spark's key-value RDD aggregation model, documented in the Spark RDD programming guide. Assuming online time is measured in minutes, 120 minutes represents two hours.

QUESTION NO: 10

In the online log query scheme, the?processing to complete the calculation work. During the whole calculation process, the intermediate calculation results need to be

For temporary storage, which of the following components are suitable for storing intermediate calculation results? (multiple choice)

A. HDfS

B. HBase

C. Kafka

D. Redis

ANSWER: C D

Explanation:

Kafka and **Redis** are appropriate temporary stores for intermediate results in an online log-query architecture because both support low-latency, distributed processing workflows. Kafka provides durable, ordered event streams that can hold calculation outputs between processing stages. Producers can write intermediate records to a topic and downstream consumers can independently read, replay, or continue processing those records. This decoupling is especially useful for continuously generated log data and supports fault-tolerant stream pipelines. Kafka's design and topic-based retention model are documented in the Apache Kafka documentation.

Redis is suitable when intermediate results must be accessed and updated very quickly, such as counters, aggregation values, session-level state, lookup data, or short-lived query results. Its in-memory data structures allow processing tasks to store and retrieve temporary state with very low latency, while expiration policies can automatically remove data after the required retention period. Redis therefore complements Kafka: Kafka carries and retains streams of intermediate events, while Redis provides fast shared state and temporary result caching. Redis data types, expiration, and persistence capabilities are described in the Redis documentation.

QUESTION NO: 11

pass throughHBaseofcreateTableThe method creates a table, what parameters must be passed in?

A. Table Name

B. table names and columns

C. Table names and column families

D. can be empty

ANSWER: C

Explanation:

Table names and column families is correct. In the HBase Java API, a table is created by submitting a table descriptor that identifies the table and includes at least one column-family descriptor. A table name is required because it uniquely identifies the HBase table in its namespace. Column families must also be defined at creation time because HBase stores data physically by column family: each family has its own storage settings and is the unit through which properties such as compression, block size, versions, and Bloom filters are configured. Individual column qualifiers, commonly called columns, do not need to be declared when the table is created. They can be supplied dynamically when data is written, provided that they belong to an existing column family. For example, current HBase APIs use `TableDescriptorBuilder.newBuilder(TableName)` together with one or more `ColumnFamilyDescriptor` instances before invoking the Admin client's `createTable` operation. This design enables schema flexibility for qualifiers while retaining explicit control over the storage-oriented column-family structure. Huawei Big Data development practices follow this standard HBase table-definition model. See the Apache HBase Admin createTable API and the HBase schema documentation.

QUESTION NO: 12

aboutKafkaThe characteristics of the following description are correct? (multiple choice)

A. KafkaIt is a high-throughput, distributed, publish-subscribe-based messaging system

B. KafkaPersistence of messages

C. KafkaApplicable to offline and online message consumption scenarios

D. Kafkaguarantee eachPartitionmessages in order

ANSWER: A B C D

Explanation:

Kafka is correctly characterized as a high-throughput, distributed publish-subscribe messaging system. It uses a distributed cluster of brokers and topic partitions, allowing producers and consumers to scale horizontally while decoupling data producers from downstream processing applications. Kafka persists messages: records appended to a topic partition are written to durable storage and retained according to configured time- or size-based retention policies. This persistence enables consumers to read data independently and, when appropriate, replay records by controlling their offsets.

Kafka is also suitable for both online and offline consumption. Online consumers can process newly produced events with low latency, while batch or offline workloads can consume retained historical data later for analytics, ETL, or reprocessing. Finally, Kafka guarantees ordering within an individual partition. Records are appended to a partition in a defined sequence and are read in that same sequence by a consumer; therefore, applications requiring ordering should route related records to the same partition, commonly by using a consistent message key. These behaviors are core Kafka design principles documented by Apache Kafka in its concepts and terminology documentation and design documentation.

QUESTION NO: 13

existFusionInsight HDclient, executeskinit{account}command is to getKDCwhich content?

A. Krb5.conf

B. TGT

C. ST

D. jaas.conf

ANSWER: B

Explanation:

The correct answer is **TGT**. In a Kerberos-enabled FusionInsight HD client environment, running `kinit {account}` authenticates the specified principal with the Key Distribution Center (KDC). After successful authentication, the KDC's Authentication Service issues a Ticket-Granting Ticket (TGT), which is saved in the client's credential cache. The TGT represents the authenticated user session and is subsequently presented to the Ticket-Granting Service when the client needs to request service tickets for protected cluster components, such as Hadoop, HBase, Hive, or other Kerberos-secured services. The command can obtain credentials through a password prompt or through keytab-based authentication when the relevant parameters are supplied. A TGT has a limited validity period, so users or automated jobs may need to renew or reacquire it before it expires. This behavior follows the Kerberos protocol's initial authentication process: the client first obtains a TGT from the KDC and then uses that ticket to obtain tickets for individual services. See the MIT Kerberos kinit documentation and RFC 4120: The Kerberos Network Authentication Service.

QUESTION NO: 14

FusionInsight HDin, belonging toStreamingWhat are the roles of the service? (multiple choice)

A. Nimbus

B. Supervisor

C. Broker

D. quorumpeer

ANSWER: A B

Explanation:

In FusionInsight HD, the Streaming service is based on Apache Storm's distributed stream-processing architecture. **Nimbus** is the master-side daemon: it accepts submitted topologies, allocates work across the cluster, distributes topology code, and monitors execution. This control-plane role enables the Streaming service to coordinate the processing topology as a whole. **Supervisor** runs on worker nodes and manages the worker processes that actually execute the assigned topology tasks. It receives assignments from Nimbus, starts or stops worker processes as required, and ensures that the node provides the execution capacity allocated to it. Together, Nimbus and Supervisor provide the central coordination and per-node execution-management roles required for a Storm-based Streaming deployment. Apache Storm's architecture documentation describes Nimbus as analogous to Hadoop's JobTracker and Supervisors as analogous to TaskTrackers, clarifying their complementary responsibilities. See the Apache Storm architecture documentation and Huawei's MapReduce Service product documentation for the FusionInsight/MRS big-data service context.

QUESTION NO: 15

existKafkamiddle,ConsumerIt can support offline message and online message consumption at the same time, but consume data sequentially

of.

A. True

B. False

ANSWER: A

Explanation:

True is correct. Kafka retains records for a configured retention period independently of whether they have already been consumed. A consumer can therefore start from an earlier stored offset to perform offline or historical processing, and then continue reading newly produced records as it catches up to the end of the log. This makes the same Kafka topic suitable for both backlog-oriented processing and ongoing, near-real-time consumption without requiring a separate data copy.

Kafka organizes each topic as one or more partitions, and records appended to an individual partition have monotonically increasing offsets. A consumer reads records from each assigned partition in offset order, providing sequential consumption for that partition. Consumer groups track committed offsets, allowing a consumer to resume after interruption and continue through retained historical messages before processing subsequent online messages. When a topic has multiple partitions, Kafka does not promise one global ordering across all partitions; the sequential ordering described here is the ordering within each partition. Huawei Kafka deployments follow this core Kafka consumer and offset model. See the Apache Kafka consumer documentation and the Apache Kafka topic and partition documentation for the retention, offset, and per-partition ordering model.

QUESTION NO: 16

In Hive, which clause is used to specify the directory location of an external table?

A. ROW FORMAT

B. LOCATION

C. PARTITIONED BY

D. CLUSTERED BY

ANSWER: B

Explanation:

`LOCATION` is correct. When an external Hive table is created, the `LOCATION` clause identifies the HDFS or compatible file-system path containing the table data. Hive stores the table metadata in the metastore while the data remains at the specified external location.

`PARTITIONED BY` defines partition columns, `CLUSTERED BY` defines bucketing, and `ROW FORMAT` defines the row serialization format. See the Apache Hive DDL Language Manual.

QUESTION NO: 17

FusionInsightHDin, aboutHivepartition (partition) function, which is wrong as described below?

A. Partition fields are defined when the table is created

B. There can only be one partition field, and multi-level partitions cannot be created

C. Using partitions can reduce the data scan range of some queries, thereby improving query efficiency

D. Partition fields can be used aswherecondition of the sentence

ANSWER: B

Explanation:

“There can only be one partition field, and multi-level partitions cannot be created” is the incorrect description. Hive partitioning is defined with one or more partition columns in the table definition. When multiple partition columns are specified, Hive organizes partition data hierarchically according to the order of those columns, commonly called multi-level

partitioning. For example, a table can be partitioned by `year`, `month`, and `day`; each year directory then contains month directories, and each month directory contains day directories. This structure enables partition pruning when a query includes predicates for partition columns, so Hive can limit the partitions and files it needs to read rather than scanning all table data. Partition columns are therefore an important design choice: they should align with frequent query filters while avoiding an excessive number of very small partitions. Hive's language manual explicitly supports declaring a comma-separated list of partition columns in `PARTITIONED BY`, and its partition-filter documentation describes pruning based on partition predicates. See the Apache Hive DDL language manual and the Apache Hive language documentation for the partition-table syntax and behavior.

QUESTION NO: 18

HiveWhich of the following table types are supported? (multiple choice)

A. Partition Table

B. bucket table

C. Skewed table

D. partition+bucket table

ANSWER: A B C D

Explanation:

Hive supports partitioned tables, bucketed tables, skewed tables, and tables that combine partitioning with bucketing. A partitioned table organizes data into separate directory paths according to one or more partition-column values. This lets Hive prune irrelevant partitions when a query filters on partition columns, reducing the data that must be read. A bucketed table divides rows into a configured number of files based on a hash of one or more bucket columns. Bucketing can support efficient sampling and can be useful when data is consistently organized for joins or aggregations. Hive also supports skewed tables, where frequently occurring values for specified columns can be stored separately to help manage highly uneven data distributions. These features are not mutually exclusive: a table may first be partitioned and then bucketed within each partition. In Hive DDL, the relevant clauses are `PARTITIONED BY`, `CLUSTERED BY ... INTO ... BUCKETS`, and `SKEWED BY`. The combined design is therefore a valid Hive table layout. See the Apache Hive DDL language manual and the Hive bucketed tables documentation.

QUESTION NO: 19

RDDasSparkThe core object, which has the following characteristics? (multiple choice)

A. read only

B. partition

C. fault tolerance

D. efficient

ANSWER: A B C D

Explanation:

Spark's core abstraction is a resilient distributed dataset, a distributed collection designed for parallel processing across cluster partitions. "read only" is correct because this abstraction is immutable: transformations do not modify an existing dataset, but instead define a new dataset derived from the prior one. "partition" is correct because data is divided into partitions, allowing tasks to process separate portions concurrently on executors. This partition-based organization is fundamental to Spark's distributed execution model and enables parallelism to be adjusted according to available resources.

"fault tolerance" is correct because Spark records the lineage of transformations used to create a dataset. If a partition is lost, Spark can recompute that partition from its lineage rather than requiring a complete copy of all intermediate data. "efficient"

is also correct in the intended Spark context: datasets can be persisted in memory or on disk, avoiding repeated computation for iterative algorithms and interactive workloads. Together, immutability, partitioning, lineage-based recovery, and persistence provide the reliable and high-performance processing model associated with Spark's core data abstraction. See the Apache Spark RDD Programming Guide and the Spark RDD API documentation.

QUESTION NO: 20

HDFSused is"Write once, read many"file access model. So it is recommended that a file be created, written and closed After closing, do not modify it again.

A. True

B. False

ANSWER: A

Explanation:

True is correct. HDFS is designed around a write-once, read-many access pattern that is optimized for storing and processing very large files across distributed cluster nodes. A client normally creates a file, writes its content sequentially, and closes it. Once closed, the file is treated as immutable: its content is not intended to be modified in place. This design simplifies distributed consistency, improves throughput for large sequential reads, and supports HDFS replication and fault-tolerance mechanisms efficiently. In practical big-data workflows, if a dataset must be changed after it has been finalized, the usual approach is to create a replacement file or write a new version of the dataset rather than edit arbitrary existing bytes within the closed file. HDFS supports appending under controlled circumstances, but append capability does not change the core recommended model stated in the question: create, write, close, and subsequently read the file without modifying it. This model is a fundamental HDFS characteristic and is important when designing data ingestion and downstream analytics pipelines. See the Apache Hadoop HDFS architecture documentation at HDFS Architecture and the Hadoop FileSystem specification at Hadoop FileSystem Specification.

QUESTION NO: 21

existFusionInsight HDofHBase, which of the following scenarios will not triggerFlushoperate?

A. When the client initiates a scan request, it willscan cachingset too large

B. Initiate by client-specified methodFlushdo

C. RegionServerTotal memory exceeds threshold

D. RegionofMemStoreset over threshold

ANSWER: A

Explanation:

When the client initiates a scan request, it willscan cachingset too large is correct because scan caching controls read-side behavior only. In HBase, the Scan caching setting specifies how many rows the client requests from a RegionServer in each scanner RPC. Raising this value can reduce the number of remote procedure calls and may affect client and server read processing, response size, and memory used while serving the scan. It does not write data into a region's MemStore, alter MemStore accounting, or request persistence of buffered mutations to HFiles. Consequently, an unusually large scan caching value does not meet any HBase flush condition.

A flush is fundamentally a write-path operation: data already accepted into an in-memory MemStore is persisted as immutable HFiles in the region's store. HBase documentation distinguishes scan caching as a client scanner optimization from MemStore management and flushing. This separation is important when diagnosing performance behavior: scan parameters should be tuned for efficient result retrieval and RPC volume, whereas flush behavior is governed by the write buffer and MemStore lifecycle. See the Apache HBase reference guide's discussion of client scan caching and its description of MemStore and flushing.

QUESTION NO: 22

FusionInsight HDWhich of the following belong toOozieofMapReduce Actionconfiguration item? (multiple choice)

A. name-node

B. source (HDFSofAction)

C. mapred.mapper.class

D. job-tracker

ANSWER: A C D

Explanation:

An Oozie MapReduce action defines the Hadoop execution environment and the MapReduce job parameters that Oozie submits. **name-node** is a core MapReduce-action element: it supplies the HDFS NameNode URI used by the action, including resolution of job resources and paths. **job-tracker** is also a core action element in the classic MapReduce/YARN-compatible Oozie action definition, identifying the Hadoop endpoint to which the job is submitted. In FusionInsight HD deployments using this action model, these values connect the workflow action to the target Hadoop cluster services.

mapred.mapper.class is a valid Hadoop MapReduce configuration property. It can be placed in the action's `configuration` section to specify the mapper implementation class for a job using the legacy `mapred.*` API. Oozie passes such configuration properties to the submitted MapReduce job, allowing job-specific mapper, reducer, input, output, and other execution settings to be defined without changing the workflow's control-flow structure. The Oozie MapReduce Action specification documents the action-level cluster elements and embedded configuration support, while Hadoop documents the MapReduce configuration model: Apache Oozie MapReduce Action Extension and Apache Hadoop MapReduce Tutorial.

QUESTION NO: 23

FusionInsght ManagerWhat interfaces are supported when interfacing with external management platforms?

(multiple choice)

A. SNMP

B. VPN

C. BGP

D. syslog

ANSWER: A D

Explanation:

FusionInsight Manager supports **SNMP** and **syslog** for integration with an external management platform. SNMP provides the standardized management interface through which a third-party network-management system can obtain monitored information and receive alarm notifications from managed resources. This lets an operations platform centrally observe status, performance-related indicators, and fault events using established SNMP management mechanisms. The SNMP protocol's management model, including notifications such as traps and informs, is defined by the IETF in RFC 3411.

Syslog is also an external-management integration interface because it enables FusionInsight Manager to forward operational and alarm log records to a centralized log server or security/operations platform. Centralized syslog collection supports consolidated event analysis, alert correlation, auditing, and retention outside the cluster-management console. The syslog message format and transport considerations are standardized in RFC 5424. Together, SNMP for management monitoring and alarms, and syslog for centralized log reporting, provide the relevant external O&M-platform connectivity described for FusionInsight Manager.

QUESTION NO: 24

existMapReduceIn application development of , which of the followingJAVAThe class is responsible for managing and running a Counting tasks?

A. Job

B. Context

C. FileSystem

D. Configuration

ANSWER: A

Explanation:

`Job` is the class used by a MapReduce application to define, configure, submit, monitor, and manage a MapReduce computation. An application creates a job instance, associates it with its configuration, and supplies the processing components and execution settings needed for the computation, such as mapper and reducer implementations, input and output formats, input and output paths, key and value classes, and reduce-task count. Once configured, the application submits the job to the Hadoop framework, typically through methods such as `waitForCompletion` or `submit`. The framework then schedules and runs the tasks required by that job and exposes status and progress information through the job object. This responsibility makes `Job` the appropriate answer for managing and running a MapReduce counting task. Hadoop's MapReduce tutorial describes job configuration and submission as the application's central interaction with the framework: Apache Hadoop MapReduce Tutorial. The API reference for the class also documents its role in representing and controlling a submitted MapReduce job: Hadoop Job API.

QUESTION NO: 25

useFusionInsight HDofHiveQuery data, when the intermediate result of the query has a large amount of data, You can choose to compress the intermediate result data for better performance. Which of the following areHiveCorrelation of intermediate result compression parameter? (multiple choice)

A. hive.exec.compress.intermediate (middle)

B. hive.intermediate.compression.codec

C. hive.exec.compress.output

D. hive.intermediate.compression.type

ANSWER: A B D

Explanation:

Hive supports compression of intermediate data produced between query-processing stages, such as map output that is transferred and consumed during subsequent processing. `hive.exec.compress.intermediate` is the switch that enables or disables this intermediate-result compression. Once compression is enabled, `hive.intermediate.compression.codec` specifies the compression codec to use, such as a Hadoop-supported codec configured in the cluster. `hive.intermediate.compression.type` controls the compression format type, commonly at record or block granularity depending on the available codec and execution framework. Together, these settings allow an administrator or developer to balance network and disk I/O reduction against the CPU cost of compression and decompression. This is especially useful for large shuffle volumes, where reducing the amount of data written, read, and transferred can improve query execution performance. These properties are documented as Hive configuration properties for intermediate compression. See the Apache Hive configuration properties reference and the Apache Hadoop compression documentation.

QUESTION NO: 26

aboutFusionInsight HDplatformHiveservice, itsWebHCatDevelopment interface, which of the following descriptions is incorrect?

A. Support based onRESTquery request

B. WebHCatThe return data format isSML

C. WebHCatbased onHTTPandHTTPSAgreement to provide services to the outside world

D. able to passWebHCatCreate tables, query, etc.

ANSWER: B

Explanation:

WebHCatThe return data format isSML is the incorrect description. WebHCat, also known as Hive Templeton, exposes Hive and related Hadoop operations through a REST-style web interface. Its API responses are represented as JSON objects, allowing HTTP clients to submit jobs, obtain job status, retrieve results or metadata, and process returned information using standard JSON parsers. “SML” is not a WebHCat response format; even if the text was intended to mean XML, XML is not the normal response representation specified for the WebHCat REST API. The Apache Hive WebHCat API documentation shows JSON response bodies for operations such as status queries and Hive query submission. This JSON-oriented interface is important for application integration because it provides a consistent, language-independent structure for response fields, status values, identifiers, and error information. WebHCat is designed to make Hive capabilities accessible to external applications without requiring those applications to directly use the Hive command-line client or native Java APIs. See the Apache Hive WebHCat documentation and the WebHCat API reference.

QUESTION NO: 27

A large-scale production enterprise plans to transform its internal logistics data and sales data into big data. The design department gives

Based on the analysis of the data storage scheme, which of the following statements are correct?

(multiple choice)

A. HBaseIt can store massive data and support dynamic expansion, which can fulfill the storage requirements of logistics and sales data.

B. HbaseIt supports efficient random reading, and can complete real-time analysis and command of the logistics situation through reasonable design.

C. Logistics data is very sparse,HBaseCan handle sparse data efficiently.

D. HBaseIt can be built on ordinary commercial servers, and the construction cost is relatively low.

ANSWER: A B C D

Explanation:

HBase is appropriate for this type of enterprise data platform because it is a distributed, column-family NoSQL database built for very large tables. Its region-based distribution and integration with the Hadoop ecosystem enable capacity and throughput to be expanded by adding nodes, making it suitable for continually growing logistics records and sales data. HBase also provides low-latency, row-key-based random reads and writes. When logistics entities, timestamps, locations, and statuses are modeled into an effective row-key and column-family design, current shipment or operational-status records can be retrieved promptly to support near-real-time monitoring and operational decisions.

The sparse-data characteristic is also an important fit. HBase stores columns only when values are present, rather than requiring every row to contain all possible attributes. This makes it efficient for logistics events whose available fields vary by event type, source system, or processing stage. Finally, HBase is designed to run as a distributed service on clusters of commodity hardware, rather than requiring specialized high-end storage systems. This supports incremental scale-out and can reduce infrastructure cost for a large internal data platform. See the Apache HBase Reference Guide and Apache HBase project documentation.

QUESTION NO: 28

In Solr, which query parameter is commonly used to specify the default field for query terms when the query does not explicitly identify a field?

A. `q`

B. `fq`

C. `df`

D. `wt`

ANSWER: C

Explanation:

`df` is correct. The `df` parameter specifies the default field used by the query parser for terms that are not qualified with a field name. For example, when `df=content` is configured, a query for `hadoop` is interpreted against the `content` field by the applicable query parser.

`q` supplies the main query expression, `fq` supplies filter queries, and `wt` selects the response writer. These parameters have different purposes from selecting the default query field. See the Apache Solr Standard Query Parser documentation.

QUESTION NO: 29

FusionInsight HDin, aboutHivethe data load function (viaHiveofLOADcommand to import data), the following description

What is wrong?

A. available fromHDFSdirectly into thehivesurface

B. available fromHiveServerThe local hard disk of the node is directly loaded into thehivesurface

C. It can be directly loaded from the local hard disk of the node where the client is located.hivesurface

D. HiveThe data loading process will not parse the specific file content, mainly the process of file transfer

ANSWER: C

Explanation:

The incorrect statement is *"It can be directly loaded from the local hard disk of the node where the client is located.hivesurface"*. In the FusionInsight HD HiveServer-based execution model, a `LOAD DATA LOCAL INPATH` operation uses a path on the local filesystem of the HiveServer node that executes the request, then transfers the file into the table's HDFS location. It is therefore not sufficient for the source file merely to exist on the filesystem of an arbitrary client node submitting the SQL. This distinction is important when users connect remotely through a JDBC client or other Hive client tool: the command is processed by HiveServer, and its local-path access is associated with that server-side execution environment. The load operation is a file movement operation; it does not parse the data records into columns during the load. For the syntax and behavior of `LOAD DATA`, including the distinction made by the `LOCAL` keyword, see the Apache Hive LanguageManual DML. Huawei FusionInsight HD documentation is available through the MapReduce Service documentation portal.

QUESTION NO: 30

FusionInsight HDin, useStreamingofLinuxWhen submitting a topology in command line mode, you need to first use an ownStreamingUser with submit permissionkinitway authentication

A. True

B. False

ANSWER: A

Explanation:

True is correct. In a security-enabled FusionInsight HD environment, submitting a Storm topology from the Linux command line is an authenticated operation. Before the submission command is run, the operating-system user must obtain a valid Kerberos ticket by using `kinit` with the credentials of a user that has been granted the required Storm topology-submission permission. The ticket establishes the user identity used by the Storm client when it communicates with the cluster's secured services, including the components involved in topology submission and resource access.

This requirement enforces the cluster's authentication and authorization model rather than relying only on the local Linux account. The identity authenticated through Kerberos must also be authorized in FusionInsight's permission-management configuration for the relevant Storm operation. Administrators commonly verify the ticket after authentication using `klist`, then submit the topology while that ticket remains valid. This protects shared clusters by ensuring that topology ownership and access decisions can be traced to an authenticated user identity. Huawei's FusionInsight/MRS documentation describes secure cluster administration and component operations at Huawei Cloud MRS documentation. The underlying Storm security model and its Kerberos-based client authentication concepts are also documented by Apache Storm Security.

QUESTION NO: 31

Which of the following scenarios is notflinkWhat does the component excel at()?

(multiple choice)

A. Batch iterative computing

B. Stream processing

C. Processing

D. Data Storage

ANSWER: A B

Explanation:

Flink is designed as a unified engine for both bounded and unbounded data workloads. Therefore, **Batch iterative computing** is an appropriate use case: bounded data can be processed through Flink's batch execution capabilities, and iterative processing has historically been supported for algorithms that repeatedly refine results. **Stream processing** is also a core Flink strength. Flink executes continuous event streams with stateful processing, event-time semantics, checkpoint-based fault tolerance, and low-latency results. These capabilities make it suitable for applications such as real-time analytics, monitoring, fraud detection, and event-driven pipelines. Apache Flink describes its runtime as supporting both stream and batch processing through one common execution model, rather than requiring separate processing engines. Its stateful stream-processing model also provides the foundation for reliable continuously running jobs. See the Apache Flink architecture overview and the Flink DataStream API overview for details on these processing capabilities.

QUESTION NO: 32

existSparkmiddle,SparkSQLis an independent module that does not depend onSparkCorefinish independentlySQLSentence parsing, optimization

operations such as transformation and execution.

A. True

B. False

ANSWER: B

Explanation:

False is correct because Spark SQL is a Spark module that operates within the broader Spark runtime rather than functioning as a completely independent processing engine. Spark SQL provides structured-data abstractions such as DataFrames and SQL, and it performs SQL parsing, analysis, logical-plan optimization, physical planning, and code generation through components including the Catalyst optimizer. However, the resulting execution plan is run by Spark's underlying execution infrastructure. Spark Core supplies the foundational distributed capabilities needed to schedule tasks, manage memory, recover from failures, communicate across the cluster, and execute computations on workers.

In practical use, a SQL statement submitted through a SparkSession is converted into an optimized query plan, then translated into executable work that Spark schedules across executors. This integration is why Spark SQL can combine SQL queries with other Spark capabilities and data-processing APIs while retaining fault tolerance and distributed execution. Huawei Big Data Developer candidates should therefore recognize Spark SQL as a high-level structured-data module built on and integrated with Spark's core engine, not as a module able to complete all parsing, optimization, transformation, and execution independently of Spark Core. See the Apache Spark SQL programming guide and the Apache Spark cluster overview.

QUESTION NO: 33

FusionInsight HD HBasein the cluster,Table1belongNamespace1,Table2belong

Namespace2,Table1There are two column families, namelycf11,c12,Table2There is a column family namedcf21,

Which of the following options will allow the user accountAalso havecf11andcf21read and write permissions. (multiple choice)

A. Assign this user accountglobalread permission

B. Give this user accountNamespace1read and write permissions

C. Assign this user accountTable1andTable2read and write permissions

D. Assign this user accountNamespace1andNamespace2read and write permissions

ANSWER: C D

Explanation:

"Assign this user accountTable1andTable2read and write permissions" is correct because HBase ACLs can be granted at table scope. Read and write privileges assigned on each table apply to the data in that table, including its column families. Therefore, granting the required permissions for both Table1 and Table2 gives the account access to cf11 in Table1 and cf21 in Table2.

"Assign this user accountNamespace1andNamespace2read and write permissions" is also correct because namespace-level authorization covers tables contained in the respective namespaces. Since Table1 belongs to Namespace1 and Table2 belongs to Namespace2, read and write permissions on both namespaces provide the needed access to the relevant column families. HBase permission scopes are hierarchical: permissions may be applied globally, at namespace level, at table level, and, where needed, at column-family or qualifier level. A namespace or table grant is appropriate here because each required column family is within a different target table and the requested access includes both reading and writing. See the Apache HBase access-control documentation and the HBase security guide.

QUESTION NO: 34

Tonganeng's societyHDofe, user-definedUDFcan andHmachine built-inUDFclass name,

In this case, the user-definedUDF.

A. True

B. False

ANSWER: A

Explanation:

True is correct. In Hive, a user-defined function can be registered using the same name as an existing built-in function. When a temporary user-defined function has that name, Hive resolves calls in the current session to the user-defined implementation, effectively allowing the custom function to override the built-in function for that session. This is useful when an application needs compatible SQL syntax but requires specialized business logic, data handling, or behavior that differs from Hive's standard implementation. The override is session-scoped when the function is created as a temporary function, so it does not permanently alter the built-in function for other users or sessions. Developers should nevertheless choose names carefully and document such overrides, because queries using the same function name can produce different results depending on whether the custom function has been registered in the active session. Hive's UDF documentation describes how custom scalar functions are implemented and registered, while its language manual documents the creation and use of temporary functions. See the Apache Hive UDF language manual and the CREATE FUNCTION documentation.

QUESTION NO: 35

An application requires simultaneous and twoFusionInsightsCluster interaction: both need to access the cluster1ofHBaseservice, need to visit

Ask the cluster2ofHiveServe;

So which of the following operations are required? (multiple choice)

A. in the cluster1Create the account required for this application on theHBaseServe

B. in the cluster2created onAThe account with the same name in the options, and set this account to be able to access the cluster'sHiveServe

C. Aoptions andBThe account created by the option must belong tosupergroupGroup

D. cluster1and cluster2Complete the mutual trust operation

ANSWER: A B D

Explanation:

The required setup is to create the application account in cluster 1 for access to the HBase service, create an account with the same name in cluster 2 and authorize it to access the Hive service, and complete mutual trust between cluster 1 and cluster 2. FusionInsight cross-cluster service access relies on consistent identity mapping: the application must have a recognized user identity in each cluster where it accesses protected services. The account in the HBase cluster supplies the authenticated identity for HBase requests, while the same-named account in the Hive cluster enables that identity to be granted the required Hive permissions.

Mutual trust is also essential because the clusters use Kerberos-based authentication. Trust configuration allows credentials issued in one cluster's security domain to be accepted and validated by services in the other cluster, enabling the application to access both services without an unrelated identity context. The account should receive only the service permissions needed by the application; cross-cluster interoperability does not require assigning it to the highly privileged `supergroup`. Huawei's MRS/FusionInsight documentation describes cluster security, user management, and Kerberos-based authentication in the Huawei Cloud MRS documentation and its User Guide.

QUESTION NO: 36

existSpark, which of the following statements about broadcast variables is correct? (multiple choice)

A. Each task saves a copy of a broadcast variable.

B. Each Executor saves a copy of a broadcast variable.

C. Broadcast variables are read-only.

D. Broadcast variables are generated through the SparkContext.broadcast() method.

ANSWER: B C D

Explanation:

“Each Executor saves a copy of a broadcast variable” is correct because Spark distributes the broadcast data to executor processes and caches it there. Tasks running in the same executor can use that local cached value, avoiding repeated shipment of the same large, read-only lookup data with every task. This is especially useful for joins or transformations that repeatedly consult relatively small reference data.

“Broadcast variables are read-only” is also correct. A broadcast variable represents a shared immutable value for use by distributed tasks. Its purpose is to provide a consistent value efficiently across executors rather than to support distributed updates. This immutability allows Spark to safely reuse the broadcasted data during task execution.

“Broadcast variables are generated through the SparkContext.broadcast() method” is correct because the driver creates a broadcast variable by passing a value to `SparkContext.broadcast(...)`. Spark then manages distribution of that value to executors when it is needed. The Spark programming guide documents broadcast variables as a mechanism for caching a read-only variable on each machine, and the Spark API documents `broadcast` as the method that creates the broadcast variable. See Apache Spark RDD Programming Guide: Broadcast Variables and PySpark SparkContext.broadcast API.

QUESTION NO: 37

FusionInsigt HD which of the following components can be used for data collection? (multiple choice)

A. Flume

B. Loader

C. HBase

D. MapReduce

ANSWER: A B

Explanation:

In FusionInsight HD, **Flume** and **Loader** can be used for data collection and ingestion. Flume is a distributed, reliable service intended to collect, aggregate, and move large volumes of streaming log or event data. Its source-channel-sink architecture lets it receive data from sources such as application logs and forward that data into the big-data storage and processing environment. This makes it suitable for continuous collection scenarios.

Loader is FusionInsight HD’s bulk data loading capability. It is used to import data from external relational databases and other supported sources into the cluster, including destinations such as Hive, HDFS, and HBase. Loader supports scheduled or batch-oriented ingestion, so it complements Flume’s streaming-oriented collection model. Together, these components cover common data-collection requirements: continuous acquisition of generated records and controlled bulk import of existing enterprise data. Huawei’s component overview identifies Flume as a log collection component and describes Loader as a tool for importing external data into FusionInsight: Huawei Cloud MRS Component Overview. Additional FusionInsight HD documentation is available through the Huawei Cloud MRS User Guide.

QUESTION NO: 38

FusionInsight HDofLoader, a connector can only be assigned to one job.

A. True

B. False

ANSWER: B

Explanation:

False is correct. In FusionInsight HDLoader, a connector is a reusable configuration object that defines the connection information and properties needed to access a particular data source or destination. Jobs reference a connector when they need to read from or write to that configured endpoint. This separation lets administrators create a connector once and use it in multiple jobs that share the same connection requirements, rather than duplicating endpoint configuration for every job. Reusing connectors also improves consistency: changes to shared connection settings can be managed centrally, while individual jobs retain their own scheduling, mapping, and execution settings. Therefore, assigning a connector to one job does not consume it or prevent another job from selecting it. Huawei's FusionInsight/MRS documentation organizes Loader administration around separately managed connections and jobs; see the Huawei Cloud MRS User Guide and the Huawei Cloud MRS documentation portal.

QUESTION NO: 39

aboutFusonInsight HDofSpark, which of the following programming languages can be used to developSparkapplication? (multiple choice)

A. C

B. Scala

C. Java

D. Python

ANSWER: B C D

Explanation:

Scala, Java, and Python can be used to develop Spark applications in a FusionInsight HD Spark environment. Spark is implemented primarily in Scala and provides native APIs that enable Scala developers to create distributed data-processing applications directly. Its Java API supports the same core distributed programming model, allowing Java applications to define RDD or DataFrame-based processing and submit jobs to the cluster. Python is supported through PySpark, which exposes Spark's distributed processing capabilities to Python developers and is commonly used for data engineering, SQL processing, and machine-learning workflows. These supported language APIs let an application define transformations and actions while Spark distributes execution across cluster resources. Huawei's FusionInsight documentation describes Spark as a distributed computing framework and provides development guidance for Spark applications; the broader Spark language API model is also documented by the Apache Spark project. See Huawei Cloud MRS Spark Development Guide and Apache Spark Documentation.