PassQueen
a.k.a dumpsqueen.com

HCIA-Big Data V3.0

Huawei H13-711_V3.0

Version Demo

Total Demo Questions: 66

Total Premium Questions: 664

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Big Data Overview	117
Topic 2, Hadoop Basic	125
Topic 3, HDFS	59
Topic 4, MapReduce	21
Topic 5, Hive	61
Topic 6, HBase	70
Topic 7, Spark	48
Topic 8, Flink	62
Topic 9, Kafka	55
Topic 10, Mix Questions	46
Total	664

QUESTION NO: 1

The compatibility of F1ink is reflected in which of the following aspects?

A. Can be mixed with Hadoop's original mappers and reducers

B. YARN can be used as a resource scheduling manager for Flink clusters

C. Ability to use Hadoop formatted input and output

D. Ability to get more data from local

ANSWER: A B C D

Explanation:

Flink is designed to integrate closely with Hadoop ecosystems, so all of the listed capabilities describe its compatibility. It can reuse Hadoop MapReduce code by running Hadoop-style mappers and reducers alongside Flink processing logic. This permits incremental migration and lets existing batch-processing assets remain useful while applications adopt Flink's execution engine. Flink also supports deployment on YARN, where YARN supplies cluster resource management and scheduling while Flink manages the application's execution components. This makes it practical to share an established Hadoop cluster with Flink workloads. In addition, Flink provides Hadoop compatibility components that allow jobs to consume and produce data through Hadoop input and output formats, enabling access to existing storage and data pipelines. Finally, Flink's execution and input-split assignment are locality-aware: where the deployment environment and input format provide locality information, work can be placed to favor data that is available locally, reducing network transfer and improving efficiency. These integrations are central to Flink's role as a framework that can operate within, rather than replace, a Hadoop-based big-data platform. See the Apache Flink Hadoop compatibility documentation and the Apache Flink YARN deployment documentation.

QUESTION NO: 2

Sparki, like Hadoop-, is not suitable for generational computinG.

A. True

B. False

ANSWER: B

Explanation:

False is correct because Apache Spark is specifically designed to support iterative and repeated computations efficiently. Unlike a purely disk-oriented batch-processing approach, Spark can cache or persist resilient distributed datasets and DataFrames in memory across multiple operations. This avoids repeatedly reading the same intermediate data from storage, which is valuable for workloads such as machine-learning training, graph processing, interactive data analysis, and iterative algorithms. Spark also provides a unified execution engine and libraries for SQL, streaming, machine learning, and graph-oriented processing, allowing these workloads to reuse data and computation plans effectively. In an iterative workflow, an algorithm commonly performs many passes over a dataset until a model converges or a calculation reaches a result. Spark's persistence mechanisms are intended to accelerate exactly this pattern, while still allowing storage levels that use disk when memory is insufficient. Therefore, it is inaccurate to state that Spark, like Hadoop, is not suitable for iterative computation. Spark's architecture is a key reason it is widely used for such workloads. See the Apache Spark RDD Persistence guide and the Apache Spark MLlib guide.

QUESTION NO: 3

Which of the following are the built-in string functions of Hive?

A. trim()

B. abs ()

C. substr()

D. length()

ANSWER: A C D

Explanation:

The built-in Hive string functions are `trim()`, `substr()`, and `length()`. `trim()` removes leading and trailing spaces from a string value, making it useful when normalizing text fields before comparisons, joins, or grouping. Hive also supports `ltrim()` and `rtrim()` when only one side of a value must be cleaned. `substr()` returns a specified portion of a string, using a start position and, optionally, a length. This is commonly used to extract elements such as prefixes, date components stored as text, or fixed-width identifiers. Hive documents `substring()` as well as `substr()` for this purpose. `length()` returns the number of characters in a string, allowing queries to validate, filter, or profile textual data. These functions are included in Hive's built-in user-defined function catalog and can be used directly in HiveQL expressions, including `SELECT`, `WHERE`, and transformation logic. The Apache Hive language manual lists these functions under string operations: Hive LanguageManual UDF. General Hive query-language documentation is available from the Apache Hive Language Manual.

QUESTION NO: 4

Indataduring stream processinG. eachthingpieceTimeWhich of the following three can be divided into?

A. Processing time, the time when the stream processing system processes the event.

B. Event time, the time when the event occurred.

C. Ingestion time, the time when the event arrives at the stream processing system.

D. Delay time, the elapsed time from event occurrence to processing.

ANSWER: A B C

Explanation:

Stream-processing systems commonly distinguish time semantics using processing time, event time, and ingestion time. Processing time is the clock time of the machine or system that executes an operation on a record. It is readily available during execution and reflects when the processing engine handled the data. Event time is assigned from the time at which the underlying real-world event happened, usually from a timestamp carried in the event itself. It enables computations such as windows and aggregations to represent the actual business sequence of events, even when records arrive out of order. Ingestion time is assigned when the stream-processing system first receives the record. It provides a system-controlled arrival timestamp, which can be useful when the producer does not provide a reliable event timestamp. These three concepts are fundamental because they distinguish the occurrence of an event, its arrival at the processing platform, and its eventual execution by that platform. Apache Flink, whose time-semantic model is widely used by stream-processing services, documents these definitions and their implications for event-time processing and watermark handling. See the Apache Flink time attributes documentation and the event-time and watermark documentation.

QUESTION NO: 5

What modules does the Spark framework include?

A. Spark SQL

B. Spark Streaming

C. Spark GraphX

D. Spark Mllib

ANSWER: A B C D

Explanation:

Apache Spark is a unified analytics engine whose ecosystem provides specialized libraries on top of its core distributed processing capabilities. **Spark SQL** is included for processing structured and semi-structured data through SQL, DataFrames, and Datasets. **Spark Streaming** is a Spark module for scalable stream processing; in modern Spark releases, its capabilities are commonly delivered through Structured Streaming. **Spark GraphX** is the graph-computation library, supplying graph abstractions and graph-parallel operations for analyzing relationships among vertices and edges. **Spark MIlib** refers to Spark's machine-learning library, which provides algorithms and utilities for common machine-learning workflows such as classification, regression, clustering, feature processing, and pipeline construction. These libraries share Spark's distributed execution model, allowing data engineers and analysts to use a consistent platform for SQL analytics, streaming workloads, graph processing, and machine learning. Apache Spark's official overview identifies SQL and DataFrames, Structured Streaming, MLlib, and GraphX as the major components of the platform. See the Apache Spark project overview and the official Spark documentation.

QUESTION NO: 6

Which of the following contents can be viewed in the Loader historical job record?

A. job status

B. Job start/run time

C. dirty data link

D. Error lines/number of files

ANSWER: A B C D

Explanation:

Loader historical job records are intended to provide both an operational summary and diagnostic evidence for completed or previously run import/export tasks. The **job status** identifies the resulting execution state, allowing an administrator to quickly determine whether processing succeeded, failed, was stopped, or remains in another recorded state. **Job start/run time** supplies the timing information needed to review when the task ran and assess its execution period. A **dirty data link** is available when applicable so that records rejected during processing can be located and investigated, supporting data-quality remediation. **Error lines/number of files** provides useful job-result statistics, including error-related information and file-processing quantities, so administrators can evaluate the outcome and scale of the operation. Together, these fields make the historical record useful for routine monitoring, troubleshooting failed or partially successful loads, and validating data-ingestion results. Huawei's MapReduce Service documentation describes Loader as a component for transferring and processing data in MRS, while its documentation portal provides the associated Loader operation and job-management guidance. See Huawei Cloud MRS Product Description and Huawei Cloud Documentation.

QUESTION NO: 7

In the Fusioninsight product, which of the following descriptions are correct about the topic of creating Kafka?

A. When creating TopicE of Kafkal, the number of Partitions must be set

B. When creating TopicE of Kafkal, the number of Partition copies must be set

C. Setting up multiple replicas can enhance the disaster tolerance of Kafka services

D. all of the aboveA. True

ANSWER: A B C D

Explanation:

When a Kafka topic is created in FusionInsight, its partition count and replication factor are fundamental topic configuration values. The partition count defines how the topic's data is divided into independently ordered logs, enabling workload

distribution across Kafka brokers and consumer parallelism. The replication factor (described here as the number of Partition copies) defines how many broker copies of each partition Kafka maintains. These values are therefore required inputs when creating a topic through the relevant management interface or command.

Configuring multiple replicas improves service disaster tolerance because each partition has redundant copies on separate brokers. If the broker hosting a partition leader becomes unavailable, an in-sync replica can be elected leader so that the topic remains available, subject to the cluster's replica and in-sync-replica configuration. Consequently, the statement *all of the aboveA. True* is also correct because every preceding topic-creation statement is correct. Kafka's official documentation describes partitions as the unit used to distribute topic data and replication as the mechanism that provides redundancy and fault tolerance. See the Apache Kafka topic documentation and the Apache Kafka replication documentation.

QUESTION NO: 8

The following statements about the reliability of Fusioninsight network security are correct:

A. Avoid high load on the business plane from blocking the cluster management channel

B. Network plane isolation to avoid management and service bandwidth preemption and mutual interference

C. Prevent external attackers from infiltrating real business data through management channels

D. Fusioninsight supports dividing the network into three levels: the cluster service plane, the cluster management plane, and the maintenance network outside the cluster, which are physically isolated from each other.

ANSWER: A B C D

Explanation:

FusionInsight network design separates traffic according to its purpose so that cluster operation remains stable, manageable, and protected. Avoiding high load on the business plane from blocking the cluster management channel is important because management traffic carries operational functions such as monitoring, configuration, orchestration, and fault handling. Keeping that traffic available enables administrators and cluster components to continue managing the environment even when service workloads are busy. Network-plane isolation also reserves bandwidth and reduces interference between service and management communication, improving predictability and reliability under load. Protecting management channels from external access is equally important because these channels can expose administrative interfaces and control operations; restricting their reachability helps protect the cluster and its business data. FusionInsight's separation of the cluster service plane, cluster management plane, and an external maintenance network implements defense in depth. Physical isolation provides particularly strong separation where required, limiting unintended traffic paths and reducing the impact of faults or attacks that originate on another network. These practices align with Huawei's broader security principle of network segmentation and controlled access to management interfaces. See Huawei Trust Center—Security and Huawei Cloud MRS documentation.

QUESTION NO: 9

In big data computing tasks, aboutI0Which of the following is an incorrect description of an intensive task?

A. CHigh PU consumption

B. I0During intensive task execution, most of the time is spent in I0deal with

C. By improving network transmission efficiency and read and write efficiency, performance can be greatly improved

D. The more tasks, the more efficient the CPU

ANSWER: A

Explanation:

"CHigh PU consumption" is the incorrect description. The wording "I0 intensive" is evidently a formatting-corrupted reference to an I/O-intensive task. An I/O-intensive big-data workload spends a significant portion of its elapsed time reading or writing

data, transferring data across the network, or waiting for storage and network operations to complete. Consequently, its primary performance constraint is the I/O path and its available throughput or latency, rather than sustained computation on processor cores. CPU usage can occur while requests are prepared, serialized, deserialized, compressed, or otherwise handled, but high CPU consumption is not the defining characteristic of this workload type. High and continuously saturated processor utilization instead characterizes a CPU-intensive workload, where computation itself is the limiting resource. This distinction is important when selecting tuning measures in distributed data platforms: I/O-intensive processing is assessed through storage, network, and data-access behavior, whereas CPU-intensive processing is assessed through processor capacity and computational efficiency. Hadoop's MapReduce documentation describes the framework's processing of input and intermediate data across distributed tasks: Apache Hadoop MapReduce Tutorial. Huawei's MapReduce Service documentation is available at Huawei Cloud MRS Documentation.

QUESTION NO: 10

With the continuous increase in the volume of big data, the requirements for the physical security of data storage are getting higher and higher, and higher requirements are also put forward for the multiple copies of data and disaster recovery mechanism.

A. True

B. False

ANSWER: A

Explanation:

True is correct. As data volumes grow, data platforms must protect increasingly valuable and concentrated storage assets against physical failures, site-level incidents, accidental loss, and service interruptions. Physical security includes the resilience of the facilities and infrastructure hosting storage systems, such as controlled access, environmental protection, power redundancy, and network and equipment reliability. These safeguards reduce the likelihood that an infrastructure event compromises stored data or its availability.

Multiple data copies and a disaster-recovery mechanism are equally fundamental because a single copy, or copies located in only one failure domain, cannot provide sufficient resilience for important big-data workloads. Replication preserves recoverable copies of data, while disaster recovery defines how services and data can be restored or switched to another protected location after a major fault. Huawei Cloud describes disaster recovery as protecting applications and data through cross-site recovery capabilities, and its storage documentation explains reliability mechanisms based on redundant data storage. These practices support the statement that increasing data scale raises requirements for both secure physical storage and robust replication and recovery design. See Huawei Cloud Storage Disaster Recovery Service overview and Huawei Cloud OBS data reliability documentation.

QUESTION NO: 11

Which of the following Redis data types is suitable for storing a ranking list with scores?

A. String

B. Hash

C. Sorted Set

D. Set

ANSWER: C

Explanation:

Sorted Set is correct. A Redis sorted set stores unique members together with score values and keeps members ordered by score. Commands such as `ZADD`, `ZRANGE`, and `ZREVRANGE` can be used to add members and retrieve ranking results. A normal Set does not maintain score-based ordering, while a Hash stores field-value pairs. See the Redis sorted sets documentation.

QUESTION NO: 12

In YRAN's tag-based scheduling, which of the following options is tagged?

A. APPMaster

B. ResourceManager

C. Container

D. NodeManager

ANSWER: D

Explanation:

NodeManager is the tagged entity in YARN tag-based scheduling. A NodeManager represents an individual worker node and reports that node's available resources and characteristics to the ResourceManager. By associating a tag with a NodeManager, the cluster identifies the corresponding node as belonging to a particular scheduling group or resource category. Applications can then request execution resources associated with that tag, enabling the scheduler to place containers on nodes that meet the intended deployment, workload-isolation, hardware, or locality requirement.

This approach is consistent with YARN's node-label scheduling model: labels are node-level attributes maintained by the cluster and used by the scheduler when deciding where an application is allowed to obtain containers. In practice, the tag describes the execution-node capability or classification rather than the application-management service or an individual allocated workload. Because the NodeManager is the YARN daemon running on and representing each worker node, it is the appropriate object to which the tag is associated. See the Apache Hadoop documentation for YARN node labels and the YARN architecture overview.

QUESTION NO: 13

Synchronizing data between partition replication in Kafka, copying data from the leader of the partition to the follower requires a thread (replicationFetcherThread) follower (a follower is equivalent to a consumer) to actively pull messages from the leader in batches, which greatly improves throughput.

A. True

B. False

ANSWER: A

Explanation:

True is correct. Kafka replicates each partition through a leader–follower model. The leader accepts producer writes and serves replication data, while each follower replica independently fetches records from its leader. On a broker hosting follower replicas, the replica-fetching mechanism—commonly described as a ReplicaFetcherThread or replication fetcher thread—issues fetch requests to the relevant leader brokers and writes the returned records to the follower's local log. In this respect, a follower uses a pull-based fetch pattern similar to a consumer, although its purpose is replica synchronization rather than application consumption.

Replication is performed in batches rather than one record at a time. A fetch request can obtain multiple available records, subject to fetch-size and wait-time settings, which amortizes network, protocol, and request-processing overhead across many messages. This batching is a key reason Kafka can sustain efficient replication throughput while maintaining ordered replica logs. The leader tracks which replicas are sufficiently caught up, and Kafka uses the in-sync replica set to support durable committed writes. Kafka's design documentation describes followers as pulling data from leaders and explains the replication model; see the Apache Kafka replication design documentation and the Kafka fetch configuration documentation.

QUESTION NO: 14

How many blocks are saved in HDFS by default in the FusionInsightHD system?

A. 3

B. 2

C. 1

D. uncertain

ANSWER: A

Explanation:

The correct answer is **3**. In FusionInsight HD, HDFS uses a default replication factor of three for newly written data blocks. In practical terms, HDFS divides a file into blocks and maintains three replicas of each block across suitable DataNodes. This default supports the distributed file system's core reliability objective: data remains available even if a DataNode or disk fails. The NameNode manages the metadata describing each block and its replica locations, while DataNodes store the physical replicas. HDFS placement logic also aims to distribute replicas across nodes and, where rack awareness is configured, across racks to improve fault tolerance and reduce the risk of data loss caused by a single hardware or rack-level failure. The replication factor is configurable for a file or directory when workload capacity, availability, or storage-efficiency requirements differ; however, the question asks for the default behavior. Therefore, three stored copies is the expected default HDFS setting in this context. Huawei FusionInsight documentation describes HDFS as the distributed storage component of the platform, and the standard Hadoop HDFS configuration documents the default `dfs.replication` value as 3. See Huawei Cloud MRS product documentation and Apache Hadoop HDFS default configuration.

QUESTION NO: 15

Regarding the relationship between Hive and other components of Hadoop. Which of the following descriptions is wrong?

A. Hive finally stores data in HDFS

B. Hive is a data warehouse tool for the Hadoop platform

C. HQL can execute tasks through MapReduce

D. Hive has a strong dependence on HBase

ANSWER: D

Explanation:

Hive has a strong dependence on HBase is the wrong description. Hive is a distributed data warehouse system that provides an SQL-like language, HiveQL, for querying and analyzing data held in Hadoop-compatible storage. Its normal architecture uses a metastore for table metadata and relies on a processing engine—historically MapReduce, and commonly Tez or Spark in modern deployments—to execute the generated query plan. Hive tables are ordinarily stored in HDFS or another filesystem supported by Hadoop's storage abstraction; HBase is not a required architectural component.

Hive can be integrated with HBase through an HBase storage handler. This integration allows HiveQL to read from and, in supported configurations, write to HBase-backed tables. That is an optional interoperability feature selected when HBase's low-latency, row-oriented access model is needed alongside Hive's SQL-based batch analytics. A Hive deployment can therefore be installed, configured, and used without HBase. Apache Hive's documentation describes Hive as a data warehouse system and details its execution and storage model, while its HBase integration documentation presents HBase support as a storage-handler integration rather than a mandatory dependency. See the Apache Hive project documentation and the Hive HBase integration guide.

QUESTION NO: 16

Sparki - like Hadoop - is not suitable for iterative computinG.

A. True

B. False

ANSWER: B

Explanation:

False is correct because Apache Spark is specifically well suited to iterative computing workloads, including machine-learning algorithms, graph processing, and repeated interactive data analysis. Iterative jobs commonly reuse the same input data across multiple processing stages or algorithm iterations. Spark can retain reusable datasets in memory by persisting or caching resilient distributed datasets (RDDs), avoiding repeated reads from storage and repeated reconstruction of intermediate results. This capability substantially reduces latency for workloads that repeatedly operate on the same data.

Spark also uses a directed acyclic graph execution model and schedules work across a cluster while recovering lost partitions through lineage. The combination of distributed execution, memory persistence, and fault-tolerant RDDs makes it effective for iterative computation at scale. Spark's official RDD programming guide describes persistence as an important tool for iterative algorithms and interactive use, and notes that cached datasets can be reused across operations. See the Apache Spark RDD persistence documentation and the Apache Spark documentation.

QUESTION NO: 17

YARN manages cluster resources through ResourceManager. What are its main functions?

A. Cluster resource scheduling

B. application management

C. log management

D. The above statement is not correct

ANSWER: A B

Explanation:

"Cluster resource scheduling" and "application management" are the principal ResourceManager responsibilities in YARN. The ResourceManager is the cluster-level authority for assigning shared resources to submitted workloads. Its Scheduler component receives resource requests, evaluates available capacity and configured scheduling policies, and allocates containers on appropriate NodeManagers. This is the resource-scheduling function that enables multiple applications and users to share a Hadoop cluster efficiently.

The ResourceManager also includes the ApplicationsManager, which accepts job or application submissions, negotiates the first container used to launch an ApplicationMaster, and maintains the application's lifecycle state. The ApplicationMaster subsequently coordinates the application's individual tasks, but it does so under the ResourceManager's cluster-wide resource-management model. Therefore, application admission and lifecycle coordination are correctly described as application management, while allocation policy and container assignment are correctly described as cluster resource scheduling. Apache's YARN architecture documentation explicitly identifies the ResourceManager's Scheduler and ApplicationsManager as its major components and explains these responsibilities. See the Apache Hadoop YARN overview and the YARN resource model documentation.

QUESTION NO: 18

In the Fusioninsight HD product, which statement about Kafka is incorrect?

A. Kafka strongly depends on Zookeeper

B. The number of instances deployed by Kafka must not be less than 2

C. Kafka server can generate messages

D. Consumer consumes messages as the client role of Kafka

ANSWER: C

Explanation:

The incorrect statement is “Kafka server can generate messages.” In Kafka’s architecture, messages (also called records or events) are created and published by producer applications. A producer is a client that sends records to a specified Kafka topic. Kafka servers, commonly called brokers, receive those producer records, persist them in topic partitions, replicate them as configured, and make them available for consumer clients to read. A broker therefore provides the storage, replication, and request-handling services that support message streaming; it is not the component that originates business messages. In a FusionInsight HD deployment using Kafka, applications or integrated services acting as producers are responsible for generating the data sent into Kafka, while the Kafka service manages that data after publication. This distinction is central to Kafka’s producer-broker-consumer model and is also important when diagnosing data-ingestion issues: message creation and publishing logic belongs on the producer side, whereas availability and retention of published records are broker responsibilities. Apache Kafka’s official introduction describes producers as publishing events to topics and brokers as the servers forming the Kafka cluster. See the Apache Kafka introduction and Kafka Producer API documentation.

QUESTION NO: 19

Which of the following operations are recorded in the Fusioninsight HD system audit log unavailable?

A. Manually clear warnings

B. Start and stop the service instance

C. delete service instance

D. Check history monitoring

ANSWER: D

Explanation:

Check history monitoring is the operation that is not recorded in the FusionInsight HD system audit log. In FusionInsight Manager, audit logs are intended to provide accountability for management actions that change the cluster or affect its operational state. They therefore focus on administrative activities such as actions against services and instances, configuration-related changes, and other managed operations performed through the platform. Historical monitoring, by contrast, is a read-only observation activity: it retrieves previously collected monitoring information and does not modify a service, instance, configuration, or the cluster state. Because it does not constitute an auditable management operation, viewing historical monitoring data is not included in the system audit-log records. This distinction is useful in operations and compliance reviews: audit records establish who performed an administrative action and when, whereas monitoring history is used to inspect service health and past metric trends. Huawei’s FusionInsight/MRS documentation describes Manager as the interface for cluster management, monitoring, and maintenance, and its operational documentation provides the relevant audit and maintenance context. See the Huawei Cloud MRS product documentation and the MRS user guide.

QUESTION NO: 20

In a point-to-point messaging system, data in the queue can be consumed by one or more consumers, but a message can only be consumedoneSecond-ratE.

A. True

B. False

ANSWER: A

Explanation:

True is correct. A point-to-point messaging model uses a queue as the destination for messages and supports competing consumers: multiple consumers may be attached to, or receive work from, the same queue. However, each individual

queued message is delivered to only one consumer rather than being broadcast to every consumer. This provides work distribution, allowing consumers to process different messages in parallel while preventing the same queued message from being processed independently by multiple recipients. After a consumer successfully receives and acknowledges a message, the messaging system treats that delivery as completed and removes the message from the queue. This one-message-to-one-consumer delivery characteristic is the defining behavior of point-to-point queue messaging. It is distinct from publish/subscribe delivery, where a message sent to a topic can be delivered to multiple subscribers. Oracle's JMS tutorial describes the point-to-point domain as using queues and specifies that each message has only one consumer: Oracle Java Message Service tutorial. The JMS specification also defines queue consumers and the queue-based messaging model: Jakarta Messaging specification.

QUESTION NO: 21

Which of the following scenarios does Hive not apply to?

A. Non-real-time analysis. Such as log analysis, statistical analysis

B. data mininG. Such as user behavior analysis, interest analysis, regional display

C. Data summary. such as clicks per user per day,Click to rank

D. Real-time online data analysis

ANSWER: D

Explanation:

Real-time online data analysis is the scenario to which Hive generally does not apply. Hive is a distributed data-warehouse system designed for querying, analyzing, and summarizing large datasets stored in Hadoop-compatible storage. Its SQL-like HiveQL statements are translated into execution jobs on processing engines such as MapReduce, Tez, or Spark. This architecture is highly effective for large-scale, offline and batch-oriented analytical workloads, where throughput and the ability to process massive historical datasets are more important than immediate response times. Because query planning, resource scheduling, and distributed job execution introduce latency, traditional Hive is not intended to provide the low-latency, continuously updated responses required by real-time online analysis. Hive is therefore commonly used to build data warehouses, generate periodic reports, aggregate logs, and support data-mining analysis over accumulated data. Huawei's Big Data training materials position Hive as an offline data-warehouse component in the Hadoop ecosystem; real-time needs are typically addressed with streaming-oriented components and architectures. For background on Hive's role as a data-warehouse system for reading, writing, and managing large datasets, see the Apache Hive project website and the Apache Hive Getting Started documentation.

QUESTION NO: 22

In HBase, which component is responsible for coordinating Region assignment and load balancing?

A. HMaster

B. HRegionServer

C. DataNode

D. ZooKeeper

ANSWER: A

Explanation:

HMaster is correct. HMaster manages the HBase cluster by assigning Regions to RegionServers, detecting RegionServer failures, coordinating recovery, and balancing Region placement. RegionServers process normal client read and write requests for the Regions assigned to them. HDFS provides persistent storage, and ZooKeeper supports coordination and service discovery. See the Apache HBase HMaster documentation.

QUESTION NO: 23

What are the key features of Streaming in Huawei's big data product Fusioninsight HD?

A. flexibility

B. Scalability

C. Disaster recovery capability

D. message reliability

ANSWER: A B C D

Explanation:

FusionInsight HD Streaming is designed to ingest, transmit, and process continuous event data in distributed big-data environments. **flexibility** is a key characteristic because streaming services can support varied data sources, topics, consumers, and real-time processing scenarios, allowing deployments to adapt as data flows and business requirements change. **Scalability** is essential because streaming workloads commonly grow in throughput, topic count, and consumer concurrency; distributed nodes and partitions allow capacity to expand horizontally. **Disaster recovery capability** is provided through distributed deployment, replication, and high-availability mechanisms that preserve service continuity and enable recovery when nodes or sites experience failures. **message reliability** is also fundamental: durable storage, replication, producer acknowledgements, and consumer offset management support dependable delivery and controlled processing of streaming records. Together, these capabilities let a platform handle real-time data at scale while maintaining availability and dependable message handling. Huawei describes FusionInsight as a big-data platform for distributed data processing and analytics; its streaming capabilities are built around these enterprise operational requirements. See the Huawei FusionInsight product information and the Huawei Cloud MRS product description for platform and component context.

QUESTION NO: 24

F1ink state preservation mainly depends on( )mechanism, which regularly backs up the state in the program.

A. Checkpoint

ANSWER: A

Explanation:

Checkpoint is correct because Apache Flink preserves the recoverable state of a streaming application by periodically creating consistent snapshots of its state. During checkpointing, Flink coordinates the distributed operators in the job so that the state belonging to a specific processing position can be recorded consistently. The generated checkpoint data is written to configured durable storage, allowing the job to restore operator state after a failure and resume processing from a consistent point. This is the foundation of Flink’s fault-tolerance model and, together with replayable input sources, supports exactly-once state consistency for appropriately configured jobs. The checkpoint interval controls how regularly these backups are initiated, while the state backend and checkpoint storage determine how and where the state snapshots are kept. Thus, the mechanism that regularly backs up program state is checkpointing. Apache Flink’s official documentation describes checkpoints as globally consistent snapshots of application state and explains their role in recovery and fault tolerance. See the Apache Flink checkpointing documentation and the Flink stateful stream processing overview.

QUESTION NO: 25

Which of the following statements about segment file in Kafka Logs is correct?

A. Mapping all index metadata to memory can avoid index data IO disk operations of segmentfile

B. The index file is sparsely stored, which can greatly reduce the space occupied by the metadata of the index file

C. Sparse storage, that is, to store the original complete data only at intervals

D. Message can be quickly located through index information

ANSWER: A B D

Explanation:

Kafka partitions their append-only logs into segment files, with each segment accompanied by index structures that associate message offsets with physical positions in the segment's log file. "Mapping all index metadata to memory can avoid index data IO disk operations of segmentfile" is correct because Kafka uses memory-mapped index files, allowing index lookups to be served through mapped memory rather than requiring application-level disk reads for each lookup. The operating system can load the required index pages efficiently.

"The index file is sparsely stored, which can greatly reduce the space occupied by the metadata of the index file" is also correct. Kafka does not create an index entry for every record; it periodically records offsets and file positions. This keeps index files compact even when a segment contains many messages. "Message can be quickly located through index information" follows from this design: Kafka finds a nearby indexed position and then scans forward in the sequential log to reach the requested offset. This combination preserves efficient sequential storage while enabling practical offset-based lookup. Kafka's log and index design is described in the Apache Kafka design documentation and its log index documentation.

QUESTION NO: 26

What are the types of znodes in Zookeeper?

A. sem1-persistent

B. ephemeral

C. temporary

D. persistent

ANSWER: B D

Explanation:

In ZooKeeper, **ephemeral** and **persistent** describe the two fundamental znode lifetime models. A persistent znode remains in the ZooKeeper data tree until a client explicitly deletes it, regardless of whether the client that created it remains connected. This makes persistent znodes suitable for durable configuration, service metadata, and coordination state that must survive individual client sessions.

An ephemeral znode is associated with the session of the client that created it. ZooKeeper automatically removes the znode when that client session expires or ends. This behavior is essential for distributed coordination patterns such as tracking active service instances, membership, and leader-election candidates, because stale registrations are removed automatically after a failed client is detected. ZooKeeper also supports sequential creation variants, which append a monotonically increasing sequence number to a znode name, but those variants are combined creation modes based on the persistent or ephemeral lifetime behavior. The Apache ZooKeeper documentation describes persistent and ephemeral znodes and their session-related behavior in its znode documentation: ZooKeeper Programmer's Guide and ZooKeeper CreateMode API.

QUESTION NO: 27

Which of the following reasons may be the cause of the failure of the Loader job execution?

A. Job execution takes too long

B. Does not comply with data conversion rules

C. No data compression format specified

D. No source directory or file specified

ANSWER: A B C D

Explanation:

All of the listed conditions can cause a Loader job to fail. **Job execution takes too long** can result in the job exceeding a configured execution-time limit or being terminated after prolonged processing. **Does not comply with data conversion rules** is a valid failure cause because Loader must parse source records and convert them according to the configured field, delimiter, and data-type conversion requirements before loading. **No data compression format specified** can prevent Loader from correctly reading compressed source data when the required compression handling is not configured or cannot be identified. **No source directory or file specified** leaves the job without an input location to read, so execution cannot proceed. These settings should be validated when a job is created: confirm that the input path exists and is accessible, conversion rules match the source-data layout, compression handling matches the actual files, and runtime limits are realistic for the input volume and cluster resources. Huawei's MRS documentation portal provides Loader-related configuration and operation guidance at Huawei Cloud MRS Documentation. Product-level context for MRS data-ingestion and processing capabilities is available at Huawei Cloud MRS.

QUESTION NO: 28

The figure below shows the data transmission architecture of flumE. What is the component at the "?" in the figure?

A. Interceptor

B. Channel processor

C. Channel selector

D. None of the aboveA. True

ANSWER: C

Explanation:

Channel selector is the correct component. In an Apache Flume agent, a source receives events and passes them to a channel processor. The channel processor applies any configured interceptors and then uses a channel selector to determine the channel or channels to which each event will be written. This makes the channel selector the routing element between source-side event handling and the configured channels. A replicating selector can send an event to every configured channel, whereas a multiplexing selector can route events to selected channels according to configured header-based rules. After events are placed in a channel, a sink removes them from that channel and sends them to the next destination. Therefore, in the standard Flume transmission architecture, the component responsible for selecting the destination channel at the indicated routing point is the channel selector. Apache Flume's official User Guide describes the channel processor as containing the interceptor chain and channel selector, and documents the available selector types and their event-routing behavior. See the Apache Flume User Guide and the Flume channel documentation.

QUESTION NO: 29

Regarding RDD, which of the following statements is wrong?

A. RDD has a lineage mechanism (Lineage)

B. RDDs are stored on disk by default

C. RDD is a read-only, partitionable distributed dataset

D. RDD is Spark's abstraction of underlying data

ANSWER: B

Explanation:

RDDs are stored on disk by default is the wrong statement. An RDD is Spark's fault-tolerant distributed collection abstraction, and Spark does not automatically write every RDD to disk merely because it has been created or used in a transformation. RDD transformations are evaluated lazily: Spark records the transformations needed to construct the dataset and performs the actual computation only when an action requires a result. If an RDD is needed again and has not been explicitly persisted, Spark can reconstruct it from its lineage rather than relying on a default disk-resident copy.

Disk storage becomes relevant when an application explicitly selects a persistence strategy, such as a disk-based `StorageLevel`, or when Spark uses disk as part of a configured storage level. Developers can call `persist()` or `cache()` to retain an RDD for reuse; the default storage level used by `persist()` is memory-only, not disk. This design lets applications choose the performance, memory-use, and fault-tolerance trade-off appropriate to their workload. Spark's documentation describes lazy RDD evaluation and persistence behavior in the RDD Programming Guide, while the available persistence choices are defined in the StorageLevel API.

QUESTION NO: 30

In order to consider performance optimization. It is recommended to deploy LdapServer and KrbServer on the same node in all clusters

A. True

B. False

ANSWER: A

Explanation:

True is correct. In a Huawei MRS cluster, LDAP-based identity information and Kerberos authentication are closely related parts of the security infrastructure. Deploying LdapServer and KrbServer together on the same management node is a recommended topology for performance optimization because authentication processing frequently requires interaction between these services. Colocation reduces network communication overhead and latency for these internal security operations, helping provide more efficient authentication and identity lookup processing for cluster services and users.

This recommendation is particularly relevant to normal production cluster planning, where the management-service layout should minimize unnecessary cross-node calls while retaining the required high-availability design. It is a deployment optimization recommendation rather than a statement that only one node may host security services: in an HA environment, corresponding service instances are deployed according to the cluster's redundancy and reliability requirements. Huawei MRS documentation describes Kerberos as the authentication mechanism used to secure access to cluster services and provides the security-management context in which these management components operate. See the Huawei Cloud MRS User Guide and the Huawei Cloud MRS product documentation.

QUESTION NO: 31

Huawei Fusioninsight HD is the first big data platform in China that complies with national financial and other garbage protection. What are the following aspects of its security?

A. system security

B. Authority authentication

C. Data Security

D. Everything above

ANSWER: D

Explanation:

Everything above is correct because FusionInsight HD security is designed as a layered capability rather than as a single control. System security protects the platform's infrastructure and operating environment through measures such as secure deployment, host and network protection, vulnerability management, and auditing. Authority authentication ensures that

users and service components are reliably identified before accessing cluster resources. In a big-data environment, this identity foundation is essential because many distributed services, jobs, and storage resources interact across the cluster. Data security then protects information throughout its lifecycle, including access control for data resources, secure transmission, encryption where applicable, and auditing of sensitive operations. Together, these areas support the security and compliance objectives expected of an enterprise big-data platform, particularly where regulated or sensitive information is processed. Huawei documentation for its big-data services describes security through identity and access management, data protection, encryption, logging, and audit capabilities, reflecting this defense-in-depth approach. See Huawei Cloud's MapReduce Service product description and the Huawei Cloud Trust Center for the related security and compliance framework.

QUESTION NO: 32

Which way is correct to load data into Hive table?

A. Load the file of the local path directly into the Hive table

B. Hive supports insert into! The method of a single recorD. So you can insert a single record directly on the command line

C. Insert result set from other table into Hive table

D. Load files on HDFS into Hive tables.

ANSWER: A C D

Explanation:

Hive supports loading existing files into a table with the `LOAD DATA` statement. When the `LOCAL` keyword is used, the source path is interpreted as a path on the client machine's local filesystem, and Hive copies the file into the table's storage location. Without `LOCAL`, the source path is an HDFS path, allowing files already stored in HDFS to be moved or copied into the target Hive table or partition. These operations are commonly used for bulk ingestion of files whose format and layout are compatible with the table definition.

Hive also supports loading the output of a query into another table through `INSERT INTO` or `INSERT OVERWRITE ... SELECT`. This enables a result set from an existing Hive table to be written into a target table, which is a core pattern for transformations, aggregations, and ETL pipelines. The Apache Hive language manual documents both `LOAD DATA` for local and HDFS sources and query-based `INSERT` operations: Apache Hive LanguageManual DML. For the semantics of local versus distributed filesystem loading, see Hive LanguageManual DML archive.

QUESTION NO: 33

Elastic SearchofSowleadCanbystoragein a variety ofstorage classtype,andthe followingwherekind of storagekindtypebranchhold?

A. Shared file system

B. Object Storage

C. HDFS

D. Local file system

ANSWER: B

Explanation:

Object Storage is the correct answer because it is not a supported primary storage medium for Elasticsearch index data. Elasticsearch requires storage with filesystem semantics that allow the node to create, update, lock, rename, and reliably synchronize Lucene index files. Object storage is instead accessed through an object API and does not provide the low-latency, random-write filesystem behavior required for active shard data paths.

Object storage can nevertheless have an important role in an Elasticsearch deployment: it is commonly used as a repository destination for snapshots. A snapshot captures cluster data in a repository so that it can be restored later for backup, migration, or disaster-recovery purposes; it is not the live location from which Elasticsearch nodes serve index shards. This distinction between active data storage and snapshot repositories is fundamental when selecting Elasticsearch storage architecture. Elastic documents the filesystem-based data-path requirements and snapshot repository model in its storage and snapshot guidance. Huawei Cloud also documents Cloud Search Service operational and backup capabilities through its product documentation.

References: Elastic: Index store settings; Elastic: Snapshot and restore; Huawei Cloud Search Service documentation.

QUESTION NO: 34

Which component controls the master/slave arbitration in HDFS?

A. ZooKeeper Failover Controller

B. NodeManager

C. ResourceManager

D. HDFS Client

ANSWER: A

Explanation:

ZooKeeper Failover Controller controls master/slave arbitration in an HDFS high-availability deployment. More precisely, an instance of the ZooKeeper Failover Controller (ZKFC) runs alongside each NameNode. It monitors the local NameNode's health and participates in ZooKeeper-based leader election to determine which NameNode owns the active role. The NameNode that successfully acquires the ZooKeeper lock becomes active and serves namespace operations, while the peer remains in standby mode. This coordination prevents two NameNodes from simultaneously acting as active, which is essential for avoiding split-brain behavior and preserving HDFS namespace consistency. If the active NameNode becomes unhealthy or unavailable, the ZKFC detects the failure and coordinates automatic failover, including fencing where necessary before the standby NameNode is promoted. Thus, ZKFC is the component responsible for the active/standby, commonly described as master/slave, arbitration process; ZooKeeper provides the shared coordination service used by that process. Apache Hadoop's HDFS high-availability guide describes the ZKFC health monitoring, ZooKeeper session, and active-election responsibilities in detail: HDFS High Availability Using the Quorum Journal Manager. See also the HDFS High Availability guide.

QUESTION NO: 35

Which of the following descriptions about the deployment of big data components on Kunpeng and x86 servers is correct?

A. No shortcomings in performance

B. Single component (eg HDFS) supports mixed deployment of Kunpeng server and x86 server

C. Supports mixed deployment of Kunpeng servers and common x86 servers in a single cluster

D. Realize the autonomous control of some equipment

ANSWER: B D

Explanation:

Single component (eg HDFS) supports mixed deployment of Kunpeng server and x86 server is correct because the big-data software stack can be built for and operated on both Kunpeng's Arm architecture and x86 architecture. This enables component-level deployment planning in which instances of a distributed component, such as HDFS, can use servers from both architectures where that component's compatibility and deployment requirements are met. Hadoop's distributed design separates storage and processing across multiple nodes, and its software ecosystem is designed to run on supported Linux

platforms rather than depend on one CPU instruction-set architecture. Huawei's Kunpeng ecosystem specifically provides software-porting and compatibility resources for open-source big-data workloads on Kunpeng platforms.

Realize the autonomous control of some equipment is also correct. Introducing Kunpeng servers lets an organization use a Huawei-developed computing platform for the applicable portion of its infrastructure, rather than requiring every server to use an external x86 platform. This supports phased migration: existing compatible x86 resources can continue to provide capacity while selected big-data equipment is transitioned to Kunpeng. Such an approach can improve supply-chain flexibility and allow independent control of the hardware selected for migrated workloads. See Huawei's Kunpeng Computing portal and the Apache Hadoop project documentation for the platform and distributed-big-data context.

QUESTION NO: 36

. According to how the data flow transfers data between two Transformations. What types of data flow can be divided into?

A. redistributing stream

B. one-to-one

C. one-to-many stream

D. distributingi flow

ANSWER: A B

Explanation:

Between two transformations in Apache Flink's execution model, data flow is classified according to whether downstream parallel tasks retain the upstream partition-to-task relationship or require data repartitioning. A *one-to-one* connection sends records from each upstream subtask to the corresponding downstream subtask. This preserves the existing partitioning and is used when the downstream operation can consume the data without redistributing it across parallel instances. A *redistributing stream* transfers records across the downstream parallel subtasks according to a redistribution strategy, such as hash partitioning by key, rebalancing, broadcasting, or custom partitioning. This type of exchange is necessary when records must be routed to different downstream tasks to satisfy an operation's partitioning or parallel-processing requirements. These two categories describe the fundamental physical data-exchange relationship used to build the job graph and execute connected transformations in a distributed Flink job. Apache Flink documents the distinction as one-to-one versus redistributing exchanges in its job and task execution model; see the Flink architecture documentation and the DataStream operator documentation.

QUESTION NO: 37

Which of the following descriptions about the deployment of big data components on Kunpeng and X86 servers is correct?

A. No shortcomings in performance

B. Single component (for example: HDFS) supports mixed deployment of Kunpeng server and X86 server

C. Supports mixed deployment of Kunpeng servers and ordinary X86 servers in a single cluster

D. Realize the autonomous control of some equipment

ANSWER: B D

Explanation:

"Single component (for example: HDFS) supports mixed deployment of Kunpeng server and X86 server" is correct because Hadoop ecosystem services are distributed across multiple nodes and use portable Java-based frameworks and standard network protocols. Subject to using compatible component versions, operating systems, and native dependencies, worker nodes based on different CPU architectures can participate in the same service. For HDFS, this means DataNodes can provide storage capacity to the same HDFS namespace while running on Kunpeng or x86 hardware; the normal HDFS replication, block reporting, and client-access mechanisms continue to govern data availability. Huawei's Kunpeng big-data

software ecosystem emphasizes adaptation and support for common open-source big-data components on Kunpeng platforms. See Huawei Kunpeng Computing and the Apache Hadoop HDFS Architecture Guide.

“Realize the autonomous control of some equipment” is also correct. Deploying supported big-data workloads on Kunpeng servers provides an ARM-based server platform within Huawei’s computing ecosystem. An organization can therefore select Kunpeng-based infrastructure for applicable nodes and workloads, reducing dependence on a single external processor platform while retaining interoperable deployment where compatibility requirements are met. This is a deployment and supply-chain autonomy benefit, rather than a claim that every layer of a heterogeneous environment must be replaced at once.

QUESTION NO: 38

The client is a user operationHDFSThe most common way. the following aboutHDFSThe description of the client issureyeswhich?

A. customerendis HDFSpart of isDeploy HDFSmustGrouppiece

B. HDFS guestThe client provides something likeshe 11'sOrderVisit by wayAsk HDFSnumber inaccording to

C. HDFSThe client is a library,IncludeHDFS filePartssysteminterface, thesecatchmouth hiddenHDFSMost of the complex in the implementationmiscellaneoussex

D. customerendcan support opening,Common operations such as read and write

ANSWER: B C D

Explanation:

HDFS clients provide the primary interfaces through which applications and users access files stored in HDFS. A client can access HDFS through command-line tools, including the `hdfs dfs` command family, which offers shell-like operations for working with distributed files and directories. This makes the statement describing shell-style command access to HDFS data correct. HDFS client functionality is also supplied as a library, principally through Hadoop’s filesystem APIs such as `FileSystem`, `FSDataInputStream`, and `FSDataOutputStream`. These APIs encapsulate the distributed details of locating NameNode metadata, finding DataNode replicas, and transferring file blocks, allowing application code to use familiar filesystem operations. Therefore, the statement that the client library exposes HDFS filesystem interfaces while hiding implementation complexity is correct. The client APIs support normal file lifecycle and stream operations, including opening files and reading or writing file contents. During writes, the client coordinates with the NameNode for block placement and then streams data to DataNodes; during reads, it obtains block locations and reads from suitable replicas. These behaviors are documented in the Apache Hadoop HDFS Architecture Guide and the FileSystem API documentation.

QUESTION NO: 39

Which of the following components does the platform architecture of Huawei's big data solution include?

A. Hadoop layer

B. GaussDB 200

C. Datafarm layer

D. FusionInsight Manager

ANSWER: A C D

Explanation:

Huawei’s big-data solution architecture is built around a data-farm foundation, a Hadoop-based distributed data platform, and centralized platform management. The **Datafarm layer** provides the large-scale infrastructure and resource foundation on which clustered big-data services operate. The **Hadoop layer** supplies the distributed storage and processing ecosystem used for big-data workloads, including core capabilities for storing, managing, and analyzing massive datasets across

multiple nodes. **FusionInsight Manager** is the management component for Huawei FusionInsight clusters. It provides centralized deployment, configuration, monitoring, alarm handling, service management, and operations support, allowing administrators to manage Hadoop ecosystem services as an integrated platform. These components therefore represent complementary parts of the platform architecture: underlying data-farm resources, the distributed Hadoop data-processing layer, and the management plane. Huawei's FusionInsight documentation describes the Manager as the centralized component for managing and monitoring cluster services, while Huawei Cloud MRS documentation confirms the platform's Hadoop ecosystem basis. See Huawei Cloud MRS documentation and Huawei Cloud MRS product description.

QUESTION NO: 40

Which of the following descriptions about Spark RDD are correct?

A. RDD is an immutable distributed data collection

B. RDD transformations are evaluated lazily

C. RDD lineage can be used for fault recovery

D. RDD data can only be stored in memory

ANSWER: A B C

Explanation:

An RDD is Spark's fundamental distributed data abstraction. It represents an immutable collection of data partitions that can be processed in parallel across a cluster. RDD transformations, such as `map` and `filter`, are evaluated lazily: Spark records the lineage required to produce the result and performs execution when an action, such as `count` or `saveAsTextFile`, is invoked.

RDD lineage also supports fault recovery. If a partition is lost, Spark can usually recompute that partition from the transformations used to create it. RDDs are not restricted to residing only in memory; they can be persisted with different storage levels and may use memory, disk, or both. See the Apache Spark RDD Programming Guide.

QUESTION NO: 41

In the process of using Flume to transmit data, in order to prevent data loss due to the restart of the Flume process. Which of the following Channel types can be used

A. Memory Channel

B. JDBC Channel

C. File Channel

D. HDES Channel

ANSWER: B C

Explanation:

JDBC Channel and **File Channel** are appropriate when Flume events must survive an agent-process restart. A Flume channel is the transactional buffer between a source and a sink, so restart resilience requires the channel to persist queued events outside the agent's heap memory. JDBC Channel stores events in a relational database through JDBC. Because the event data and its transactional state are held in durable database storage, unconsumed events can be recovered when the Flume agent starts again. File Channel provides the same essential protection through an on-disk write-ahead log and data files. Its transaction model records puts and takes durably, enabling events that were accepted but not yet delivered to remain available after a normal agent restart or recovery. These persistent-channel choices support the reliable, transactional source-to-channel-to-sink flow that Flume uses for event delivery. The Apache Flume User Guide identifies File Channel as a durable channel implementation and documents JDBC Channel as a database-backed channel implementation. See the Apache Flume File Channel documentation and the Apache Flume JDBC Channel documentation.

QUESTION NO: 42

kafka Distributed message delivery is based on reliable message queues. Which of the following two main message delivery patterns are included?

A. Distribution delivery mode

B. Polling handover model

C. Publish-subscribe model

D. Point-to-point model

ANSWER: C D

Explanation:

The correct message-delivery patterns are the **publish-subscribe model** and the **point-to-point model**. Kafka is built around topics, producers, and consumers, which enables publish-subscribe delivery: producers write records to a topic, and separate consumer groups can independently subscribe to and process the same topic data. Each consumer group maintains its own offsets, so one group's consumption does not prevent another group from receiving and processing those records. This makes the model well suited to distributing the same event stream to multiple independent applications, such as monitoring, analytics, and notification services.

Kafka also provides point-to-point, queue-like processing through consumer groups. Within one consumer group, a topic partition is assigned to only one active consumer at a time. Consequently, records from that partition are processed by a single group member, allowing work to be distributed across consumers for parallel and scalable processing. Kafka's retention and offset mechanisms provide reliable replay and recovery capabilities while preserving this delivery behavior. Apache Kafka describes consumer groups and partition assignment in its consumer design documentation; its consumer introduction also explains how consumers and groups read topic partitions.

QUESTION NO: 43

In Streaming, exactly one message reliability level is achieved through the ACK mechanism.

A. True

B. False

ANSWER: B

Explanation:

False is correct. In a streaming topology, the ACK mechanism tracks whether a tuple and the tuples derived from it have been successfully processed. If processing is not acknowledged before the configured timeout, the source can replay the tuple. This acknowledgement-and-replay behavior provides an *at-least-once* processing guarantee: a message is not silently lost merely because processing failed or an acknowledgement was not returned. However, a replay can cause the same message to be processed again, so ACK tracking alone cannot ensure that each message is processed exactly once. Exactly-once semantics require additional coordination, such as transactional processing, idempotent writes, or durable state and deduplication capable of preventing repeated effects when a replay occurs. Apache Storm's reliability model, which underlies the ACK/failed-tuple approach used by streaming topologies, documents that its basic mechanism guarantees at-least-once processing and describes transactional topologies separately for exactly-once effects. See the Apache Storm message-processing guarantees documentation and the Apache Storm transactional topologies documentation.

QUESTION NO: 44

Which of the following statements about Transformation in Flink is correct?

A. The time window can be set by window

B. The Filter operation is to perform a Boolean function on each element

C. FlatMap can segment text

D. KeyBy is to group the source data by key,To ensure that the metadata of the same key is divided into the same group

ANSWER: A B C D

Explanation:

All listed statements describe valid Apache Flink DataStream transformations. The `window` operation is used after data has been logically partitioned, typically with `keyBy`, to define windows such as tumbling, sliding, or session windows. A time-based window is configured by applying a time window assigner, for example through `window(TumblingEventTimeWindows.of(...))`. The statement that the Filter operation is to perform a Boolean function on each element is correct because a filter applies a predicate to every input record and retains records for which that predicate evaluates to true. FlatMap can segment text because it is designed to produce zero, one, or many output records from one input record; the standard word-count example uses FlatMap to split each input text line into individual words. KeyBy is to group the source data by key,To ensure that the metadata of the same key is divided into the same group is also correct in principle: `keyBy` logically partitions a stream by a key so that records with identical keys are handled by the same keyed partition, enabling keyed state and keyed window operations. See the Apache Flink DataStream operators overview and the Flink window documentation.

QUESTION NO: 45

Which of the followingOSWhich version is recommended for building a Fusioninsight V1R2C60 cluster?

A. SUSE11 SP1/SP2/SP3 for AMD64&Inter64

B. CentOS6.6

C. redhat-6.4-x86_64

D. RedHat-6.5-x86_64

E. RedHat-6.7-x86_64F Ubuntu6.3

ANSWER: A B C D

Explanation:

For FusionInsight V1R2C60, the recommended operating-system baseline includes the supported 64-bit SUSE Linux Enterprise Server 11 service-pack releases and the specified Red Hat Enterprise Linux and CentOS 6 releases. Accordingly, *SUSE11 SP1/SP2/SP3 for AMD64&Inter64*, *CentOS6.6*, *redhat-6.4-x86_64*, and *RedHat-6.5-x86_64* are valid platform choices for cluster hosts. Using an operating system listed in the release's compatibility requirements matters because FusionInsight installation components, service scripts, kernel behavior, package dependencies, and high-availability tooling are validated against those exact distributions and versions. The architecture must also be 64-bit, as indicated by the AMD64/x86_64 designations; this is essential for the memory capacity and native binaries required by a distributed big-data cluster. Before deployment, administrators should also verify the detailed hardware, network, time-synchronization, and prerequisite-package requirements in the matching FusionInsight release documentation, since supported operating systems are only one part of the installation baseline. Huawei's product documentation portal is available at Huawei FusionInsight documentation, and Huawei Cloud documents the related operational concepts and cluster-management guidance at Huawei Cloud MRS documentation.

QUESTION NO: 46

Which of the following descriptions about the features of Zookeeper is wrong?

A. Updates sent by the client are applied in the order in which they were sent

B. A message is to be received by more than half-respected servers,he will be able to successfully write to disk

C. A message update can only succeed or fail. There is no intermediate state

D. The number of Zookeeper nodes must be odd

ANSWER: D

Explanation:

The statement **"The number of Zookeeper nodes must be odd"** is the wrong description. ZooKeeper does not impose an odd-number requirement on an ensemble; it can technically operate with any number of voting servers, provided it can form a quorum. What matters for consistency and availability is the quorum rule: a proposed change needs acknowledgement from more than half of the voting members. An odd number is therefore a deployment best practice rather than a mandatory feature. For example, a three-server ensemble tolerates one server failure, while a four-server ensemble also tolerates only one failure because both require three servers for a quorum. Adding the fourth server increases infrastructure cost but does not increase the number of failures the ensemble can tolerate. This is why production ZooKeeper deployments are commonly sized as three, five, or seven voting servers. ZooKeeper's architecture relies on majority quorums to ensure that two conflicting majorities cannot exist simultaneously, preserving a consistent ordered view of data. The Apache ZooKeeper administrator documentation discusses ensemble sizing and quorum behavior at ZooKeeper Administrator's Guide; its consistency guarantees are also described in the ZooKeeper Programmer's Guide.

QUESTION NO: 47

Existing server.channelsa=ch1, set the Channel type to File Channel, which of the following configurations is correct?

A. server.channels.ch1.type = file

B. server.channels.chl.type memory

C. server.channels.type memory

D. server.channels.type file

ANSWER: A

Explanation:

`server.channels.ch1.type = file` is the correct configuration because Apache Flume channel properties are defined under the agent name, the `channels` component group, and the individual channel's configured name. With `ch1` declared as a channel, `server.channels.ch1` identifies that specific channel, and the `type` property selects its implementation. Setting this value to `file` creates a File Channel, which persists events to disk and supports recovery after a process failure, making it appropriate where stronger delivery reliability is required. The surrounding Flume configuration convention uses the form `<agent-name>.channels.<channel-name>.<property> = <value>`; therefore, the channel identifier `ch1` must be included in the property path. The File Channel is documented as a durable channel implementation and requires its channel type to be explicitly set to `file`. See the Apache Flume File Channel documentation and the Flume configuration reference.

QUESTION NO: 48

Which of the following statements about the interaction between F1ink and other components is correct?

A. The implementation of Flink's checkpoint relies on Zookeeper

B. Flink can send received components to Kafka

C. The running of Flink tasks relies on YARN for resource scheduling and management

D. Flink reads and writes data in HDFS file system

ANSWER: B C D

Explanation:

Flink integrates directly with common big-data ecosystem components. “Flink can send received components to Kafka” describes Flink’s ability to use Kafka connectors: a Flink job can consume records from Kafka topics and can publish processed records to Kafka through a Kafka sink. This supports stream-ingestion, transformation, and downstream delivery pipelines. “The running of Flink tasks relies on YARN for resource scheduling and management” is correct for a Flink-on-YARN deployment. In that deployment model, YARN allocates containers and manages cluster resources for the Flink application, while Flink schedules work within the resources assigned to it. “Flink reads and writes data in HDFS file system” is also correct. Flink’s file-system abstraction supports Hadoop-compatible file systems, including HDFS, enabling jobs to use HDFS files as sources or sinks and to store durable data such as checkpoints or savepoints when configured appropriately. These integrations allow Flink to participate in a typical Hadoop-based analytics platform while retaining its own runtime and state-processing capabilities. See the Apache Flink Kafka connector documentation and the Apache Flink YARN deployment documentation.

QUESTION NO: 49

In the MRS platform, which component does the F1ume data flow not need to pass through in the node?

A. Sink

B. Channel

C. LTopic

D. Source

ANSWER: C

Explanation:

LTopic is correct. In MRS, Flume follows its standard agent-based event flow: a *Source* receives events from an external producer, a *Channel* temporarily buffers those events within the agent, and a *Sink* removes events from the channel and delivers them to the next destination or agent. These are the core components through which a Flume event is handled in an agent node. “LTopic” is not a standard Flume agent component or required stage in this flow. A topic can be relevant when Flume integrates with a messaging platform such as Kafka, because a Kafka sink may publish events to a Kafka topic; however, that is a destination-specific integration detail rather than a mandatory internal component of every Flume node’s data path. Huawei MRS provides Flume as a data collection service, while the component’s source-channel-sink processing model remains the Apache Flume architecture. See the Apache Flume User Guide for the agent data-flow model and the Huawei Cloud MRS Product Overview for MRS service context.

QUESTION NO: 50

Which of the following statements about atomicity in Zooeeper's key features is correct?

A. Updates sent by the client are applied in the order in which they were sent

B. Updates can only be fully completed or failed, not partially

C. OneA message received by one server will be received by all servers

D. No matter which server in the cluster, the same view is displayed to the outside world

ANSWER: B

Explanation:

Updates can only be fully completed or failed, not partially is correct because atomicity means that each ZooKeeper update transaction is treated as one indivisible operation. When a client submits a write request, ZooKeeper either commits the complete state change or does not apply it at all. There is no externally visible intermediate state in which only part of a transaction has taken effect. This property is essential for coordination systems, because clients can safely make decisions

based on ZooKeeper data without needing to account for partially written updates. ZooKeeper's data model and transaction processing ensure that mutations are applied consistently as complete operations, including transactions containing multiple suboperations. If processing cannot complete successfully, the requested change is not committed as a partial result. The Apache ZooKeeper documentation explicitly identifies atomicity as a core consistency guarantee: updates succeed or fail as a whole. This allows applications to use znodes and associated metadata reliably for distributed configuration, leader-election coordination, locks, and other shared-control data. See the Apache ZooKeeper Programmer's Guide and the ZooKeeper Overview for the documented guarantees and architecture.

QUESTION NO: 51

What processes are included in the HBase service of Fusioninsight HD?

A. HMaster

B. Slave

C. HRegionServer

D. Data Node

ANSWER: A C

Explanation:

The HBase service in FusionInsight HD includes the **HMaster** and **HRegionServer** processes. HMaster is the HBase control-plane process: it manages the assignment of regions to servers, helps balance regions across the cluster, coordinates schema operations such as table creation or deletion, and monitors RegionServer availability. This centralized coordination enables the HBase cluster to maintain a consistent view of where table regions are hosted.

HRegionServer is the data-serving process. It hosts and manages HBase regions, handles client read and write requests for the regions it serves, performs region splitting when regions grow, and works with the underlying distributed storage layer for durable data persistence. In a FusionInsight HD deployment, these two processes provide the essential management and data-access roles required by HBase. The Apache HBase architecture documentation describes the respective Master and RegionServer responsibilities in detail: HBase Architecture Overview. Huawei Cloud's FusionInsight documentation also identifies HBase as a distributed NoSQL database service within the big-data platform: MapReduce Service Product Overview.

QUESTION NO: 52

In Huawei's big data solution b, which of the following components are included in the bottom layer of hadoop?

A. fink

B. hive

C. miner

D. spark

ANSWER: A B D

Explanation:

The components included are **fink**, **hive**, and **spark**. In Huawei's Hadoop-based big-data platform, these components provide complementary distributed data-processing capabilities on the foundational Hadoop cluster services. Flink is used for distributed stream processing and can also process batch workloads, allowing low-latency computation over continuously arriving data. Hive supplies a data-warehouse and SQL-oriented analysis capability, enabling users to query large datasets stored in the Hadoop environment using familiar SQL syntax. Spark is a general-purpose distributed processing engine that supports batch processing, SQL analytics, streaming, machine learning, and graph-oriented workloads. Together, these engines and services form key parts of the Hadoop ecosystem exposed by Huawei big-data solutions: storage and cluster

resources are supplied by the underlying Hadoop platform, while Flink, Hive, and Spark make those resources usable for real-time processing, data warehousing, and broad in-memory analytics. Huawei's MapReduce Service documentation lists Flink, Hive, and Spark among its supported open-source big-data components and describes their roles in cluster data analytics. See Huawei Cloud MRS Product Overview and Huawei Cloud MRS Component Overview.

QUESTION NO: 53

The UNION ALL operator in Hive is used to combine the result sets of two more SELECT statements. Duplicate values are not allowed in the result set.

A. True

B. False

ANSWER: B

Explanation:

False is correct because Hive `UNION ALL` combines (concatenates) the rows returned by two or more `SELECT` queries and retains duplicate rows. This behavior is important in big-data processing when every input record must be preserved, including repeated events, transactions, or log entries that have identical selected column values. The input queries must return the same number of columns, and the corresponding columns must have compatible types; Hive then returns the combined rows without performing duplicate elimination. For example, if the first query returns a row with a particular value and the second query returns that same row, both copies occur in the `UNION ALL` output. Hive documents `UNION ALL` as the form of set operation that keeps all rows, whereas duplicate-removing behavior is associated with the distinct form of union. See the Apache Hive language manual for the syntax and semantics of union operations: Hive LanguageManual Union. The same all-rows set-operation behavior is also described in the Hive documentation hosted by Cloudera: Hive UNION syntax and behavior.

QUESTION NO: 54

The HDFS data reading process includes the following steps, please choose the correct order. (Drag picture title, sort question)

A. After obtaining this input stream, the client calls the read method to read the dat The input stream selects the nearest DataNode to establish a connection and read dat

B. The client calls closE. to close the input stream.

C. The location where the data block corresponding to this file in NameNodel is obtained by calling NameNode remotely through RPC

D. If the end of the data block has been reache Then close the connection with this DataNodel, and then re-find E. a data block. until all data is rea

E. The client calls the open method of the FileSystem instance to obtain the input stream corresponding to the fil

ANSWER: A B C D E

Explanation:

The correct HDFS read sequence starts when "The client calls the open method of the FileSystem instance to obtain the input stream corresponding to the fil". Opening the file causes the client-side HDFS implementation to request block-location metadata from the NameNode using RPC, represented by "The location where the data block corresponding to this file in NameNodel is obtained by calling NameNode remotely through RPC". The NameNode supplies locations for the file's blocks; it does not send the file's data itself. Once the stream has block locations, "After obtaining this input stream, the client calls the read method to read the dat The input stream selects the nearest DataNode to establish a connection and read dat". This locality-aware choice helps minimize network distance and improves read efficiency. When a block is consumed, "If the end of the data block has been reache Then close the connection with this DataNodel, and then re-find E. a data block. until all data is rea" describes moving to a DataNode holding the next block and repeating the read process. Finally,

“The client calls closE. to close the input stream.” releases the stream and associated network resources. This is the standard HDFS client read flow described in the Apache Hadoop HDFS Architecture Guide and the FileSystem API documentation.

QUESTION NO: 55

What is the module used to manage the active and standby states of the Loader Server process in Loader?

A. ResourcesManager

B. HAManager

C. Job Manager

D. Job Scheduler

ANSWER: B

Explanation:

HAManager is the Loader module responsible for managing the active and standby states of the Loader Server process. In a high-availability Loader deployment, Loader Server instances are arranged so that one instance provides the active service while another is available as standby protection. HAManager monitors and coordinates this active/standby relationship, enabling the service role to be maintained or switched as needed to support continuity when an active process becomes unavailable. This responsibility is specifically part of Loader’s high-availability control mechanism, rather than the functions that schedule import jobs, execute jobs, or manage general resources. By maintaining the Loader Server role state, HAManager helps ensure that Loader services remain available for data-loading operations in a clustered big-data environment. Huawei documentation describes Loader as a component for efficiently importing data into the platform and documents its deployment and HA-related administration within the MapReduce Service documentation. See the Huawei Cloud MRS Loader user documentation and the MapReduce Service product documentation for Loader context and service architecture information.

QUESTION NO: 56

Which of the following options does the time operation type supported by F1ink include?

A. End Time

B. processing time

C. Ingestion time

D. event time

ANSWER: B C D

Explanation:

Flink defines three time characteristics for interpreting timestamps and driving time-based stream operations: processing time, ingestion time, and event time. Processing time is the system time of the machine on which an operator processes a record. It is available without timestamps embedded in the data and is suitable where low latency and simple time handling are the main requirements. Ingestion time assigns a timestamp when a record enters the Flink source, providing a consistent source-side timestamp that is then carried through downstream operations. Event time uses the timestamp at which the event actually occurred in the originating system. This is the preferred model for event-driven analytics because it can correctly process records that arrive late or out of order when paired with watermark handling. Therefore, “processing time,” “Ingestion time,” and “event time” are all supported Flink time-operation types. Flink’s time concepts and their use in windowing are documented in the Apache Flink time characteristics documentation and the Apache Flink event-time and watermark documentation.

QUESTION NO: 57

In an MRS cluster, which of the following components does Spark mainly interact with?

A. Zookeeper

B. YARN

C. Hive

D. HDFS

ANSWER: B

Explanation:

YARN is correct because, in an MRS cluster, Spark commonly uses YARN as its cluster manager for distributed application execution. When a Spark application is submitted in YARN mode, Spark communicates with YARN's ResourceManager to request the resources needed by the application. YARN then allocates containers on suitable cluster nodes, in which Spark executors can be launched. This lets YARN centrally schedule CPU and memory resources, monitor application execution, and coordinate Spark workloads with other big-data processing jobs running in the same MRS environment.

This integration is fundamental to Spark's operation as a distributed compute engine in an MRS deployment: the Spark driver submits and coordinates the application, while YARN provides cluster-level resource management and container scheduling. Huawei MRS includes YARN as the resource-management layer for supported ecosystem components, including Spark. For additional details, see Huawei Cloud's MapReduce Service product documentation and Apache Spark's Running Spark on YARN guide.

QUESTION NO: 58

Which parts of the data need to be read to execute the HBase data reading business?

A. HLog

B. MemStore

C. HFile

D. HMaster

ANSWER: B C

Explanation:

HBase reads must consider both **MemStore** and **HFile**, because a region's current data can exist in either its in-memory write buffer or its persisted store files. When a client reads a row from a RegionServer, HBase checks the MemStore for recently written updates that have not yet been flushed to disk. This ensures read operations can return the newest committed values immediately, including values written after the most recent flush. HBase also reads HFiles, which are immutable, sorted files stored in the underlying distributed file system after MemStore contents are flushed. HFiles contain the durable historical data for the region and are searched using their internal indexing and filtering structures. The read result is formed by reconciling applicable cells from these sources according to HBase versioning, timestamps, and delete markers, so that the client receives the correct latest view of the row. This MemStore-plus-HFile read path is fundamental to HBase's LSM-tree-style storage design. See the Apache HBase architecture overview and its region and storage architecture documentation.

QUESTION NO: 59

Redis adopts a non-central self-organizing structurE. Nodes use the Gossip protocol to exchange node status information.

A. TRUE

B. FALSE

ANSWER: A

Explanation:

TRUE is correct. Redis Cluster is designed as a decentralized, self-organizing distributed system rather than one managed by a permanent central coordinator. Each cluster node maintains knowledge of the cluster configuration, including the hash slots it serves and information about peer nodes. This design lets the cluster continue operating and converge on membership and failure information through communication among its nodes.

Redis Cluster uses a gossip-based mechanism over its dedicated cluster bus. Nodes periodically send `PING`, `PONG`, and related cluster messages to peers, carrying information about other nodes as well as themselves. As this information is propagated, nodes learn about additions, removals, reachability, role changes, and slot ownership. The distributed exchange of status information supports automatic discovery and failure detection without requiring every state update to be sent directly to every node or relying on a single centralized management node.

The Redis Cluster specification explicitly describes the cluster bus and gossip section used to exchange information among nodes, while the Redis documentation explains the decentralized cluster model and its node-to-node communication behavior. See the Redis Cluster specification and Redis Cluster management documentation.

QUESTION NO: 60

In the MRS interface, Loader can specify a variety of different data sources, configuration data cleaning and conversion steps, configure cluster storage systems, etC. .

A. TRUE

B. FALSE

ANSWER: A

Explanation:

TRUE is correct. In Huawei Cloud MapReduce Service (MRS), Loader is a visual data-import and migration tool that lets an administrator define a data-loading job through the MRS management interface. A Loader job identifies the source from which data is obtained, including supported external or cloud data sources, and defines the destination in the MRS cluster. During job configuration, Loader provides transformation capabilities that can be used to clean, filter, map, or convert data into a format appropriate for the target system. The job configuration also includes target storage and related parameters, enabling data to be written to the applicable MRS storage or component service, such as HDFS, Hive, HBase, or other supported destinations. This visual, configurable workflow is intended to reduce the need to manually develop migration scripts while allowing users to control the source, processing rules, and storage destination for each import task. Huawei describes MRS as a managed big-data platform and documents its data migration and management capabilities in the MRS service documentation and product materials. See Huawei Cloud MRS product page and Huawei Cloud MRS documentation.

QUESTION NO: 61

When a MapKeduce program runs, the Map? ?malfunction. Which of the following options describe this task correctly?

A. AppMacter no longer starts

B. App Marter starts again

C. The task stops immediately

D. The task can still be run

ANSWER: B D

Explanation:

"App Marter starts again" and "The task can still be run" are correct. In a MapReduce job running on YARN, the ApplicationMaster is responsible for coordinating the application: it requests containers, schedules task attempts, monitors their progress, and manages recovery. If the ApplicationMaster process fails, YARN's ResourceManager can launch a replacement ApplicationMaster as a new application attempt, subject to the application's configured maximum number of attempts. This recovery behavior prevents a transient ApplicationMaster failure from necessarily ending the entire submitted MapReduce job. The replacement ApplicationMaster re-establishes coordination for the job and can continue execution. Map task output that was successfully completed and remains available can be recovered for use by later phases; work that cannot be recovered may be scheduled again. Consequently, the job is able to continue running after the ApplicationMaster restart, rather than treating the first failure as an automatic final termination. This behavior is part of YARN's application-attempt and fault-tolerance model. See the Apache Hadoop documentation on ResourceManager restart and work-preserving recovery and the MapReduce tutorial for MapReduce execution and retry concepts.

QUESTION NO: 62

Which of the following commands are of type set?

A. scard

B. sunion

C. zcount

D. hexists

ANSWER: A B

Explanation:

`scard` and `sunion` are Redis commands that operate on the Set data type. A Redis set is an unordered collection of unique string members, and these commands use set-oriented semantics. `scard` returns the cardinality of a set—that is, the number of distinct members currently stored in the specified key. This is useful when an application needs to count unique entities, such as unique visitors, tags, or permissions. `sunion` computes the union of one or more sets and returns every member that appears in at least one of the specified sets, with duplicates naturally eliminated by the set model. This supports common operations such as combining groups of users or aggregating labels across several sources. Redis documents both commands in its Set command group and identifies their time complexity in terms of the number of set members processed. See the official Redis command references for SCARD and SUNION.

QUESTION NO: 63

The ZKFC process is deployed on the following node in HDFS?

A. Active NameNode

B. Standby NameNode

C. DataNode

D. All of the above are wrong

ANSWER: A B

Explanation:

In an HDFS high-availability deployment, ZooKeeper Failover Controller (ZKFC) is installed and started on **each NameNode host**. Therefore, both **Active NameNode** and **Standby NameNode** are correct. ZKFC monitors the local NameNode's health, participates in ZooKeeper-based leader election, and coordinates automatic failover if the active NameNode becomes unavailable. Running a controller alongside each NameNode allows either NameNode to be elected to the active role when required. During failover, the controller for the former active node helps ensure that it is fenced, preventing split-

brain behavior in which two NameNodes could attempt to operate as active simultaneously. The controller associated with the standby NameNode can then promote its local NameNode to active after the required ZooKeeper election and fencing steps complete. This design is a core part of automatic failover in HDFS HA and is configured through the NameNode-side HA services rather than as a DataNode daemon. Apache's HDFS HA documentation explicitly states that a ZKFC process runs on each NameNode machine and is responsible for health monitoring and failover management. See the Apache HDFS High Availability guide and the Apache automatic failover documentation.

QUESTION NO: 64

Which of the following options is correct for the description of Compactiong?

A. There are two types of Minor and Major

B. Compation reduces the number of HFile files by merging files
CCompation reduces the number of HFile files by eliminating expired data files

C. Minor trigger frequency is higher than Major

ANSWER: A

Explanation:

There are two types of Minor and Major is correct because HBase compaction is conventionally divided into minor compaction and major compaction. HBase stores data in immutable HFiles, and compaction combines these files to maintain efficient reads and manage the storage layout. A minor compaction merges a selected subset of HFiles, typically reducing the number of smaller files without necessarily rewriting every file in a store. A major compaction rewrites all HFiles in a store into a new HFile and is therefore a more comprehensive maintenance operation. This distinction is important when operating HBase on a big-data platform: the two compaction types have different scopes, resource impacts, and operational scheduling considerations. The Apache HBase reference guide describes compactions as the process through which HFiles are selected and rewritten, including the distinction between minor and major compactions. See the Apache HBase Compaction documentation and the Apache HBase Store architecture documentation for the HFile storage and compaction model.

QUESTION NO: 65

Which of the following options are Huawei's data center solutions?

A. Data Lake Governance Center DGC

B. Cloud operating system FusionSphere Openstack

C. AI development platform ModelArts

D. Video Analysis VAS

ANSWER: A B C D

Explanation:

Huawei's data-center and cloud portfolio spans infrastructure software, data governance, artificial-intelligence development, and intelligent video capabilities. **Data Lake Governance Center DGC** is Huawei Cloud's service for organizing, governing, and managing data throughout its lifecycle, making it part of the platform's data-management solution set. **Cloud operating system FusionSphere Openstack** is a Huawei cloud operating system based on OpenStack that provides the virtualization and cloud-management capabilities used to build and operate cloud data centers. **AI development platform ModelArts** supplies an end-to-end development and operations environment for machine-learning workloads, including model development, training, deployment, and management. **Video Analysis VAS** provides cloud-based video processing and analysis capabilities that support intelligent video applications. Together, these offerings address complementary layers and workloads in a modern Huawei cloud/data-center environment: cloud infrastructure, governed data, artificial intelligence, and video intelligence. Huawei Cloud's product documentation describes DataArts Studio/DGC data governance services and

ModelArts AI development capabilities in its service catalog and documentation. See Huawei Cloud DataArts Studio and Huawei Cloud ModelArts.

QUESTION NO: 66

Fusionin, a graphical health inspection tool The sight Tool consists of FusionCare and SysCheckerp.

A. True

B. False

ANSWER: A

Explanation:

The statement is true. In the FusionInsight ecosystem, the FusionInsight Tool is used for graphical health inspection and is composed of FusionCare and SysChecker. These complementary components support inspection of the FusionInsight deployment and its underlying environment, helping administrators identify health risks and configuration issues before they affect big-data service availability. FusionCare provides the graphical inspection-oriented capability, presenting health-check information in a form that is easier for operations personnel to review and act on. SysChecker supports the associated system-level checking work needed to assess the environment on which FusionInsight runs. Together, they form the health-inspection toolset referenced by the statement. This design is consistent with FusionInsight's operational-management focus: a big-data platform must be monitored not only at the service level, but also for host and environment conditions that can affect cluster reliability. Huawei's FusionInsight product information is available at Huawei FusionInsight, and official product documentation and support resources can be accessed through Huawei Enterprise Support.