Hadoop and Spark Interview Questions with Answers

# Hadoop

## Mention Hadoop distribution?

Popular Hadoop distributions are Apache Hadoop (open-source), Cloudera Distribution of Hadoop (CDH), Hortonworks Data Platform (HDP), and Cloudera Data Platform (CDP).

## Difference between CDH and CDP

CDH (Cloudera Distribution of Hadoop) was the earlier distribution containing Hadoop ecosystem components. CDP (Cloudera Data Platform) is the modern platform after Cloudera and Hortonworks merger, integrating big data with cloud-native features, hybrid-cloud support, better security, governance, and ML/AI integration.

## Explain Hadoop Architecture

Hadoop has a Master-Slave architecture:  
- HDFS: Namenode (master) manages metadata, Datanodes (slaves) store actual data blocks.  
- YARN: ResourceManager (master) allocates cluster resources; NodeManagers (slaves) manage resources on nodes.  
- MapReduce: JobTracker (master, in Hadoop 1) or ApplicationMaster (in YARN) manages tasks; TaskTrackers or containers run tasks.

## Configuration files used during hadoop installation

Core-site.xml (core configuration), hdfs-site.xml (HDFS configs), yarn-site.xml (YARN configs), mapred-site.xml (MapReduce configs).

## Difference between Hadoop fs and hdfs dfs

Both are shell commands:  
- `hadoop fs` is generic and works with multiple file systems.  
- `hdfs dfs` is specific to HDFS operations.

## Difference between Hadoop 2 and Hadoop 3

Hadoop 3 introduced Erasure Coding for storage efficiency, support for multiple Namenodes, YARN Timeline Service v2, containerization, improved fault tolerance, and default block size of 128MB.

## What is replication factor? why its important

Replication factor is the number of copies of each data block stored across Datanodes. Default is 3. It ensures fault tolerance and data availability.

## What if Datanode fails?

HDFS automatically replicates lost blocks to other available nodes. The system continues serving data using other replicas.

## What if Namenode fails?

Single point of failure (SPOF). To solve, use Secondary Namenode, Checkpoint Node, or High Availability setup with Active-Standby Namenodes.

## Why is block size 128 MB ? what if I increase or decrease the block size

Larger block size reduces metadata load on Namenode and improves sequential read performance. Smaller block sizes cause metadata overhead and small file problem. Increasing block size may reduce parallelism.

## Small file problem

HDFS is not designed for millions of small files as each file's metadata is stored in Namenode memory, causing memory overhead. Solutions: Hadoop Archives, sequence files, or HBase.

## What is Rack awareness?

Hadoop ensures replicas are stored on different racks to improve fault tolerance and reduce data loss in case of rack failure.

## What is SPOF? how its resolved?

Single Point of Failure: In Hadoop, Namenode was SPOF. Resolved using High Availability (HA) with multiple Namenodes in Active-Standby setup.

## Explain zookeeper?

Zookeeper is a centralized service for maintaining configuration, naming, synchronization, and providing distributed coordination in Hadoop ecosystem.

## Difference between -put and -CopyFromLocal?

`-put` and `-copyFromLocal` are same, but `-copyFromLocal` explicitly copies from local filesystem. `-put` can also work with multiple sources.

## What is erasure coding?

Technique in Hadoop 3 to reduce storage overhead. Instead of replicating 3 times, it breaks data into fragments and encodes them with parity blocks, saving ~50% storage.

## What is speculative execution?

Hadoop runs duplicate tasks on slower nodes. The result from the task that finishes first is accepted, improving job completion time.

## Explain Yarn Architecture

YARN has ResourceManager (global resource allocation), NodeManagers (node-level resource management), and ApplicationMasters (per-application execution).

## How does Applications Manager and Application Master differ

Application Manager (in ResourceManager) negotiates resources. Application Master (per job) manages execution of tasks within allocated resources.

## Explain Mapreduce working?

Map phase splits input into key-value pairs. Reduce phase aggregates/summarizes output of maps. Data is shuffled and sorted between phases.

## How many mappers are created for 1 GB file?

Depends on block size. With 128MB block size, 1GB / 128MB = 8 mappers.

## How many reducers are created for 1 GB file?

Reducers are not dependent on file size but set by job configuration (default is 1).

## What is combiner? How does it work and provide performance gain? Where did you use it

Combiner is a mini-reducer that runs after the map phase to reduce data transfer during shuffle. Used for aggregation tasks like sum, count.

## What is partitioner? How does it work and provide performance gain? Where did you use it

Partitioner decides which reducer a key-value pair goes to. Default is hash partitioning. Custom partitioner can optimize data distribution and avoid skew.

# Spark

## Advantages of spark over MapReduce

In-memory computation, faster processing, supports batch & streaming, better APIs, supports MLlib, GraphX, SQL.

## Describe the architecture of Spark

Driver coordinates execution, Cluster Manager allocates resources, Executors run tasks. DAG Scheduler and Task Scheduler optimize execution.

## Yarn architecture

YARN has ResourceManager, NodeManager, ApplicationMaster. Spark integrates with YARN for cluster resource management.

## What is a cluster manager? Which ones have you used?

Cluster manager allocates resources. Types: YARN, Mesos, Kubernetes, Standalone. Commonly used: YARN, Standalone.

## Difference between SparkContext and SparkSession

SparkContext is the entry point for RDD-based APIs. SparkSession (introduced in Spark 2.0) unifies SparkContext, SQLContext, and HiveContext.

## Describe spark modes to execute the program.

Local mode (single JVM), Standalone cluster mode, YARN cluster/client mode, Kubernetes mode.

## What is dataframe and RDD. When do you use RDD over Dataframe.

RDD is low-level, immutable distributed collection. DataFrame is higher-level API with schema. Use RDD for fine-grained transformations, DF for optimized queries.

## Transformation vs Action

Transformation builds a logical plan (lazy). Action triggers execution (collect, count, save).

## Narrow transformation vs Wide transformation

Narrow: data from one partition used (map, filter). Wide: shuffle across partitions (groupByKey, reduceByKey).

## What is lazy evaluation

Spark delays execution until an action is called, allowing optimization.

## Difference between map and flapmap

map → 1 input = 1 output. flatMap → 1 input = 0 or many outputs.

## What is DAG?

Directed Acyclic Graph of transformations representing execution plan.

## What is lineage?

Record of transformations on an RDD. Used for recomputation on failure.

## Difference between DAG and Lineage?

DAG is the full execution plan. Lineage is recovery information for RDDs.

## What happens when you submit a spark job

Job → divided into stages (based on shuffle) → divided into tasks → executed on executors.

## Client mode vs cluster mode ? when to use?

Client: Driver runs on client machine (good for debugging). Cluster: Driver runs inside cluster (production use).

## Difference between a DF and a DS

Dataset (DS) is typed, DataFrame is untyped (Dataset[Row]). DS supports compile-time safety.

## Difference between a Pandas DF and a Spark DF

Pandas: single machine, limited memory. Spark: distributed across cluster.

## Coalesce vs repartition ? when to use ?

Coalesce reduces partitions without shuffle (efficient). Repartition increases/decreases with shuffle (balanced).

## If Coalesce and repartition can reduce the partitions then which one will you use?

Prefer coalesce (no shuffle) unless even distribution is needed.

## Scenario when you need to reduce the partitions?

When dataset is small and too many tasks overhead (e.g., after filtering).

## When do you need to increase the partitions?

For large datasets to enable parallelism and avoid OOM errors.

## What is a driver? Example of methods that are executed on driver?

Driver runs main() method and coordinates execution. Actions like collect(), count() return results to driver.

## What is an executor? Example of methods that are executed on executor?

Executor runs tasks. Transformations like map(), filter() run on executors.

## When would you use a broadcast join?

When one dataset is small enough to fit in memory, broadcast to all executors to avoid shuffles.

## What is a broadcast variable? How does it work and gives performance benefit.

Broadcast variable sends read-only copy of data to all executors, reducing serialization overhead.

## Cache v/s persist

Cache = persist with MEMORY\_ONLY. Persist allows different storage levels.

## What’s a shuffle?

Data redistribution across partitions during wide transformations like groupByKey.

## What is Spill? How can we use this to increase performance.

Spill = when memory is insufficient, Spark writes data to disk. Optimized by tuning memory and using compression.

## Mention different ways for Spark performance tuning. Share use case, how you identified the problem, what is the problem, what is the solution.

Techniques: caching, partition tuning, avoiding shuffles, using broadcast joins, choosing correct file formats. Example: fixed data skew with salting.

## Challenges faced in spark projects you worked on?

Data skew, OOM errors, small file issues, tuning shuffle performance.

## What is OOM error ? what are the possible reasons ?

OutOfMemory error. Reasons: insufficient memory allocation, skewed partitions, large joins, caching too much data.

## Difference between data partition and table partition?

Data partition = Spark divides dataset into chunks. Table partition = Hive/SQL-level division for faster query.

## If both the dataset are large then how do you optimize the code?

Use bucketing, repartitioning, join optimizations, avoid shuffles, consider broadcasting if possible.

## What is a logical plan vs a physical plan?

Logical plan = abstract transformations. Physical plan = optimized execution strategy (stages and tasks).

## What is accumulator?

Accumulator = write-only shared variable for aggregations (e.g., counters).

## Spark Streaming vs Structured Streaming

Spark Streaming = micro-batch, RDD-based. Structured Streaming = DataFrame/Dataset API, event-time handling, better optimization.

## What is Dynamic Partition Pruning?

Optimization where partition filters are determined at runtime, reducing scanned data.

## Advantages n disadvantages of big data File formats parquet, avro, csv, json.

Parquet: columnar, compressed, efficient. Avro: row-based, good for serialization. CSV: simple, large storage. JSON: flexible, but verbose.

## what are compression formats and its specialities

Snappy (fast, moderate compression), Gzip (high compression, slow), LZO (splittable, moderate).

## Spark optimization techniques. Share use case

Caching, coalesce/repartition, broadcast joins, vectorized I/O. Example: reduced shuffle by repartitioning before join.

## How does Spark memory management works?

Divided into Execution Memory (for shuffles, joins) and Storage Memory (for caching). Unified Memory Manager adjusts dynamically.

## How many stages and task are created.

Stages = based on shuffle boundaries. Tasks = per partition per stage.

## How are executors created in spark. What are the methods to identify executor size.

Executors created by cluster manager. Size tuned via spark.executor.memory, spark.executor.cores, spark.executor.instances.

## Explain spark-submit common parameters?

Examples: --master, --deploy-mode, --executor-memory, --num-executors, --executor-cores, --conf.

## What is data skew? How do you fix it?

Uneven distribution of data causes some tasks to take longer. Fix with salting, repartitioning, or skew join optimization.

## What is key salting? Use case and program

Adding random prefix to keys before join to distribute skewed keys. Example: key+randomInt.

## What is Adaptive Query Execution?

Feature in Spark 3 that dynamically optimizes query plans at runtime (e.g., reducing partitions, changing join strategy).

## For 1 GB file how many partitions will be created. ? support your answer with practical

Default HDFS block = 128MB → 1GB / 128MB = 8 partitions.

## For any given program, how many jobs , stages and task are created.

Jobs = triggered by actions, Stages = per shuffle boundary, Tasks = per partition per stage.

## Role of checkpointing in spark and spark streaming.

Checkpointing saves RDD lineage or streaming state to HDFS. Used for recovery and fault tolerance.