1. Why isn’t it necessary to store block locations persistently in the Name Node?

The Name Node does not need to store block locations persistently because this information is dynamically reconstructed when the HDFS (Hadoop Distributed File System) cluster starts. The Data Nodes, which store the data blocks, send block reports to the Name Node at startup, informing it of the blocks they have. This allows the Name Node to rebuild its mapping of file blocks to Data Nodes without needing to persist this information, which would otherwise make the system less flexible and more complex to maintain.

2. Why is it important to make the Name Node resilient to failures?

The Name Node is the most critical component in HDFS as it stores the metadata of the entire file system, including the directory structure, file-to-block mappings, and access permissions. If the Name Node fails, the entire HDFS becomes inaccessible because clients cannot locate files or navigate the directory structure. Therefore, making the Name Node resilient to failures is crucial for maintaining the availability and reliability of the HDFS.

3. What details are there in the FsImage file?

The FsImage file contains a snapshot of the entire file system metadata at a certain point in time. This includes:

- The complete namespace of the HDFS, including directories and file names.
- Mapping of files to the list of blocks that make up the files.
- File properties such as replication level, permissions, timestamps, etc.

FsImage is used by the Name Node to load the file system's metadata into memory during startup.

4. What is the purpose of the Secondary Name Node?

The Secondary Name Node is responsible for periodically taking checkpoints of the HDFS metadata stored by the Name Node. It does this by merging the Name Node’s in-memory edit log with the FsImage to create a new FsImage. This prevents the edit log from growing indefinitely and ensures that the Name Node can recover quickly by loading a recent checkpoint in case of a failure.

5. Does the Name Node stay in safe mode until all under-replicated files are fully replicated? Why or why not?

No, the Name Node does not stay in safe mode until all under-replicated files are fully replicated. Safe mode is a temporary state during the Name Node’s startup when it waits for a certain percentage of blocks to be reported by the Data Nodes.

The Name Node exits safe mode once a minimum number of replicas (defined by a threshold) of each block have been reported. Fully replicating all under-replicated files can take a significant amount of time, so the Name Node exits safe mode before full replication is achieved to allow the cluster to resume normal operations as soon as possible.

6. What are the core changes in Hadoop 2.x compared to Hadoop 1.x?

The core changes in Hadoop 2.x compared to Hadoop 1.x include:

- **YARN (Yet Another Resource Negotiator):** In Hadoop 2.x, YARN was introduced to decouple the resource management and job scheduling functions from the MapReduce engine, making the system more flexible and able to support multiple types of distributed applications.
- **HDFS Federation:** HDFS Federation allows multiple Name Nodes to manage separate namespaces, enabling horizontal scalability and improved performance.
- **High Availability (HA) for Name Node:** Hadoop 2.x introduced the capability for having active and standby Name Nodes to prevent single points of failure, ensuring high availability.
- **Resource Management:** YARN enables better resource management by allowing for more granular control over resource allocation and management across different applications.

7. What is the difference between MR1 in Hadoop 1.0 and MR2 in Hadoop 2.0?

- **MR1 (MapReduce 1.0):** In Hadoop 1.0, the MapReduce framework handled both the resource management and the job scheduling/execution. This tight coupling meant that Hadoop was limited to running MapReduce jobs only, and resource management was less flexible.
- **MR2 (MapReduce 2.0):** In Hadoop 2.0, MapReduce is just one of the applications running on top of YARN. YARN handles resource management and job scheduling separately, allowing Hadoop to support a wider range of distributed computing models beyond just MapReduce. MR2 also benefits from the improved scalability and flexibility offered by YARN.

8. What is HDFS Federation? What advantage does it provide?

HDFS Federation is a feature introduced in Hadoop 2.x that allows multiple Name Nodes and namespaces to exist within a single HDFS cluster. Each Name Node manages a portion of the namespace and its associated blocks, operating independently of other Name Nodes.

Advantages of HDFS Federation:

- **Scalability:** It allows the HDFS to scale horizontally by adding more Name Nodes, avoiding the bottleneck of a single Name Node.
- **Performance:** It improves performance by reducing the load on individual Name Nodes, as different Name Nodes handle different parts of the namespace.
- **Isolation:** Different namespaces can be managed independently, which can be useful for multi-tenant environments.

9. What is Name Node High Availability and how is it achieved in Hadoop 2?

Name Node High Availability (HA) in Hadoop 2 is a feature that ensures the HDFS remains operational even if the primary (active) Name Node fails.

How it's achieved:

- **Active-Standby Architecture:** There are two Name Nodes in the HA setup, an active Name Node that handles all client requests and a standby Name Node that acts as a backup.
- **Shared Edit Log:** Both Name Nodes share an edit log, usually stored in a shared storage system like NFS or a distributed file system. The standby Name Node constantly monitors this log and keeps its state synchronized with the active Name Node.
- **Failover Mechanism:** If the active Name Node fails, the standby Name Node automatically takes over, becoming the new active Name Node. This failover process can be automated using tools like Zookeeper.

10. What is the role of Application Master in YARN application execution?

The Application Master is responsible for the overall execution of an application in YARN. Its role includes:

- **Resource Negotiation:** The Application Master negotiates resources from the Resource Manager on behalf of the application. It requests the necessary amount of CPU, memory, and other resources required to execute the application.
- **Task Management:** The Application Master launches and monitors the tasks (e.g., Map and Reduce tasks in a MapReduce job) that constitute the application. It handles task scheduling, task failure recovery, and task status tracking.
- **Coordination:** The Application Master coordinates the execution of the application's tasks, ensuring that tasks are executed in the correct order and that data dependencies are managed.

- **Communication:** It communicates with the Resource Manager to manage resources and with the Node Managers where tasks are running to monitor and control task execution.