Explain the importance of below 4 demons in job execution with minimum of 5 points • Name node • Data node • Resource Manager • Node manager

Name Node:

The Name node is the Master in HDFS.

The namenode manages the filesystem namespace.

It maintains the filesystem tree and metadata for all the files and the directories in the tree.

This information is stored persistently in the local disk as two files namely : the namespace image and the editlog.

The namenode also knows the datanodes on which all the blocks for the given file are located; however it does not store the block locations persistently, because the information is reconstructed from the datanode when the system starts.

Data node:

Datanodes are the workhorses of the filesystem.

A **DataNode** stores data in the [HadoopFileSystem]. A functional filesystem has more than one **DataNode**, with data replicated across them.

On startup, a **DataNode**connects to the NameNode; spinning until that service comes up. It then responds to requests from the NameNode for filesystem operations.

They store and retrieve blocks as and when told to(by the client or the namenode)and report back to the namenode periodically with the list of blocks that they are sorting.

Client applications can talk directly to a DataNode, once the [NameNode](https://wiki.apache.org/hadoop/NameNode) has provided the location of the data. Similarly, [MapReduce](https://wiki.apache.org/hadoop/MapReduce) operations farmed out to [TaskTracker](https://wiki.apache.org/hadoop/TaskTracker) instances near a DataNode, talk directly to the DataNode to access the files.

[TaskTracker](https://wiki.apache.org/hadoop/TaskTracker) instances can, indeed should, be deployed on the same servers that host DataNode instances, so that[MapReduce](https://wiki.apache.org/hadoop/MapReduce) operations are performed close to the data.

Resource Manager:

**ResourceManager (RM)** is the master that arbitrates all the available cluster resources and thus helps manage the distributed applications running on the YARN system.

It works together with the per-node **NodeManagers (NMs)** and the per-application **ApplicationMasters (AMs)**.

In YARN, the ResourceManager is primarily limited to scheduling i.e. only arbitrating available resources in the system among the competing applications and not concerning itself with per-application state management.

Because of this clear separation of responsibilities coupled with the modularity , and with the powerful scheduler API , RM is able to address the most important design requirements – scalability, support for alternate programming paradigms.

To allow for different policy constraints, the scheduler in the RM is pluggable and allows for different algorithms. CapacityScheduler in RM is that schedules containers based on capacity guarantees and queues.

Node Manager:

The NodeManager (NM) is YARN’s per-node agent, and takes care of the individual compute nodes in a Hadoop cluster.

This includes keeping up-to date with the ResourceManager (RM), overseeing containers’ life-cycle management; monitoring resource usage (memory, CPU) of individual containers, tracking node-health, log’s management and auxiliary services which may be exploited by different YARN applications.

In YARN, the NodeManager is primarily limited to managing abstract containers i.e. only processes corresponding to a container and not concerning itself with per-application state management like MapReduce tasks.

It also does away with the notion of named slots like map and reduce slots.

Because of this clear separation of responsibilities coupled with the modular architecture, NM can scale much more easily and its code is much more maintainable.