HADOOP ARCHITECTURE

Hadoop Version 1

The basic idea is to move computing close to data, to access them efficiently and work optimally.

The basic structure is Master/Slave, there is a single JobTracker (that assigns tasks) in the cluster and a TaskTracker (that executes tasks) in each node, Figure 1 (the Job Tracker can be in the node that contains the NameNode or not).

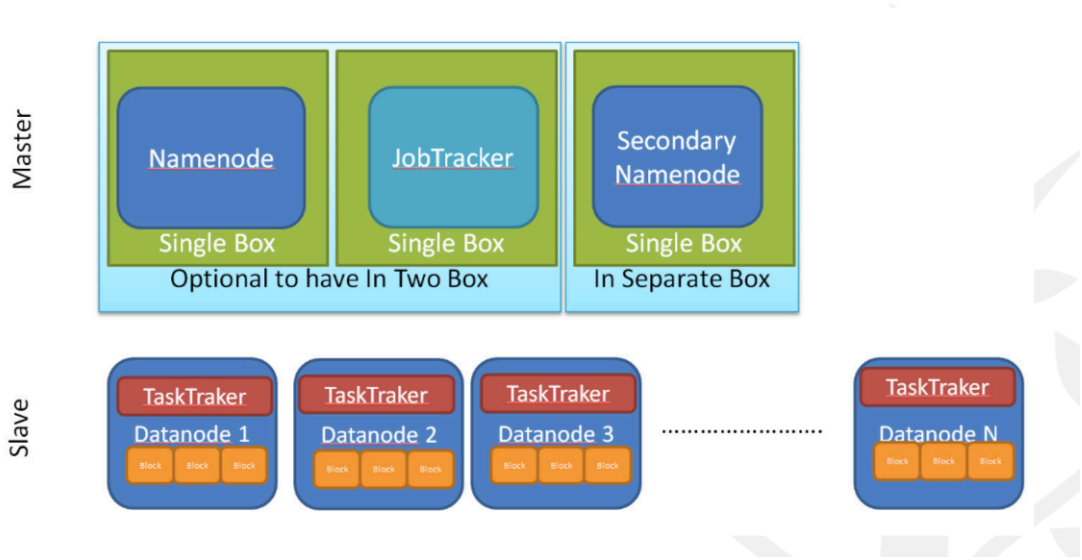

Figure 1. Architecture of Hadoop V.1

The workflow is as follows, Figure 2: The JobTracker receives a work request from the client and is responsible for distributing and monitoring the MapReduce work in the TaskTrakers, therefore, distributes the work in a balanced way across all the nodes of the cluster. The TaskTrakers of the nodes are responsible for executing the work and communicating with the JobTracker. To execute the work it must contact the Datanode to know what blocks of data must process (as close as possible to the node) to complete the work.

It is essential to monitor the work of the TaskTrackers by the JobTracker in case there are failures and another node must execute what has failed or could not be executed.

Resource allocation and job sequencing occurs internally within the framework.

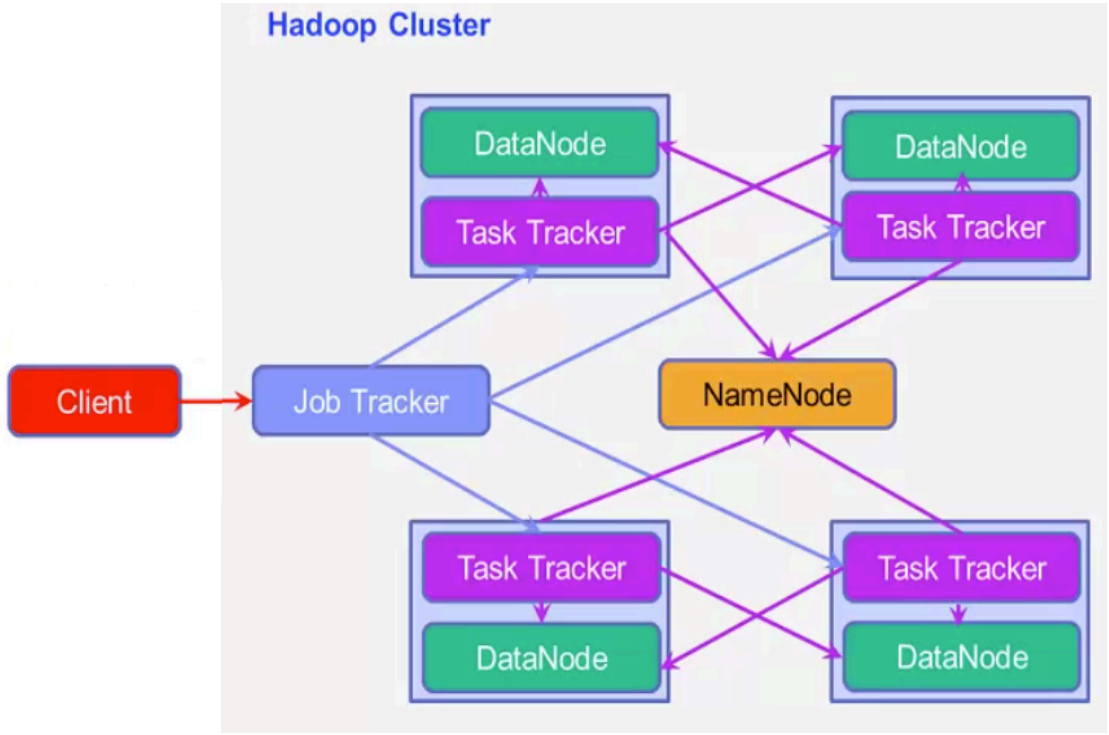

Figure 2. Work flow in the Hadoop V.1 architecture

Hadoop Version 2: YARN (Yet Another Resource Negociator) Version 1

As the version 1 architecture is designed, only MapReduce jobs one-on-one can be executed, but Hadoop can be used for more than just MapReduce jobs (it could be used for any other computation that needs data distribution in a cluster) Therefore the basic idea is to take the allocation of resources and the sequencing of the work outside the framework itself, so that it can be used for several jobs at once and, even, so that it can be used by several frameworks at once (Hadoop and Spark, for example).

On the other hand, a more efficient sequencing of the work must take place, since balancing the workload between all the nodes of the cluster may not be the most optimal depending on the work to be performed, it may not be necessary to carry the work to all cluster nodes.

The ResourceManager replaces the JobTracker but is located outside the framework and the NodeManager replaces the TaskTrackers on each node, Figure 3, but their functions are different. The NodeManager is responsible for each node of the container or containers assigned to it, so it must monitor the use of its resources (CPU, memory, network, etc.) and report it to the ResourceManager.

As in the version 1 of Hadoop, the Master node contains the NameNode, which will be consulted by the ResourceManager, and the Slave nodes will contain the DataNode.

When an application is launched by the client, the ResourceManager, step 1 of Figure 3, is contacted, which assigns it to one of the nodes to run inside a container (which has a series of resources assigned as memory, CPU, etc. depending on how YARN has been configured), steps 2a and 2b of Figure 3, if the job needs more resources to be able to complete (several mappers and reducers, several mappers and a single reduce or something completely different that does not have nothing to do with a usual MapReduce process), the node contacts the ResourceManager again and the necessary resources are negotiated to be able to execute the work, step 3 of figure 3. With YARN the ResourceManager knows about the work capacity of each node through communication with the NodeManager that is running within each node.

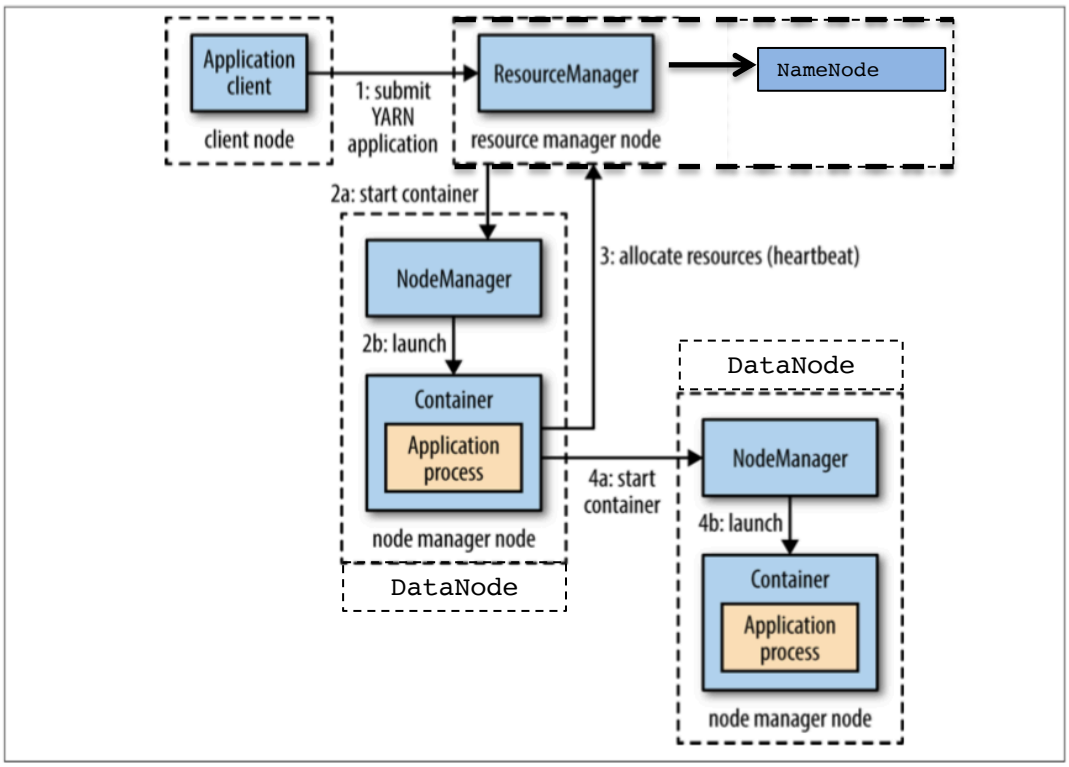

Figure 3. Hadoop V.2 architecture (YARN)

If more resources are needed, the necessary containers are assigned to complete the work in a distributed manner, step 4 of Figure 3.

To make it clearer, Figure 4 shows a workflow example of two requests that are made at the same time to the ResourceManager. These requests (Application Master –App. Mstr.- in Figure 4) are launched in two different nodes.

As can be seen, each job may need a different work capacity, that is, the execution of a number of different containers, and these containers will be executed in the most optimal nodes for it (with greater capacity, with greater block of data necessary for the work, ...), The ResourceManager is the piece of architecture that controls which nodes have capacity for new jobs, this information is provided by each of the NodeManager of each node.

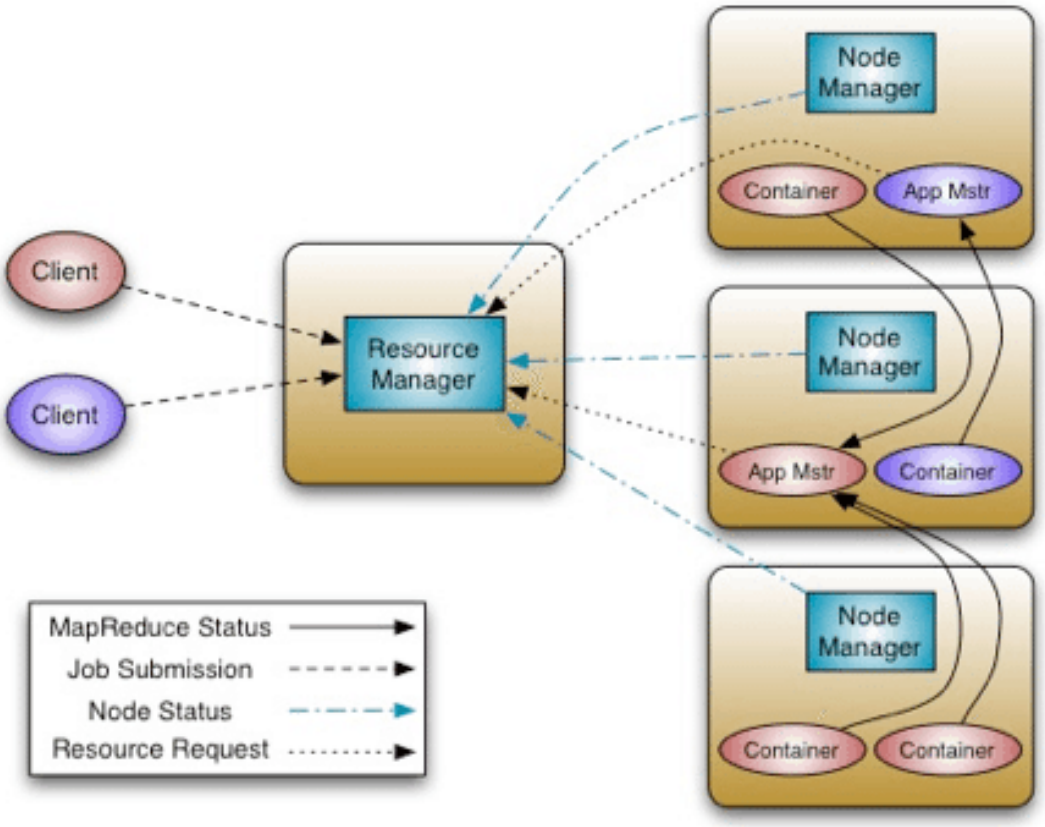

Figure 4. Workflow example for two tasks in Hadoop's YARN architecture.

Hadoop Version 3: Version 2

The main differences with version 2 of Hadoop are:

-For HDFS:

The main difference between Hadoop V.2 and Hadoop V.3 is that the fault tolerance in Hadoop V.2 is provided by the replication technique where each block of information is copied to create 2 replicas. This means that instead of storing 1 piece of information, Hadoop 2 stores three times more. This raises the problem of wasting the disk space.

In Hadoop V.3 the fault tolerance is provided by the erasure coding. This method allows recovering a block of information using the other block and the parity block. Hadoop 3 creates one parity block on every two blocks of data (using a method similar to file compression). This requires only 1,5 times more disk space. The level of fault tolerance in Hadoop V.3 remains the same, but less disk space is required for its operations.

- To run MapReduce jobs:

There are several significant changes that improve usability and scalability:

• YARN 2 supports flows - YARN application logical groups.

• Separation between collection processes (data writing) and service processes (data reading) improves scalability.