Week 06

Topics:

1. **Difference between Hadoop 1 and Hadoop 2**
2. **Hadoop YARN**

Difference between Hadoop 1 and Hadoop 2

- **Components**
- **Daemons**
- **Working**
- **Limitations**
- **Ecosystem**
- **Windows Support**

Components:

- In Hadoop 1 we have MapReduce but Hadoop 2 has YARN(Yet Another Resource Negotiator) and MapReduce version 2.

Hadoop 1	**Hadoop 2**
HDFS	**HDFS**
Map Reduce	**YARN / MRv2**

Daemons

Hadoop 1	Hadoop 2
Namenode	Namenode
Datanode	Datanode
Secondary Namenode	Secondary Namenode
Job Tracker	Resource Manager
Task Tracker	Node Manager

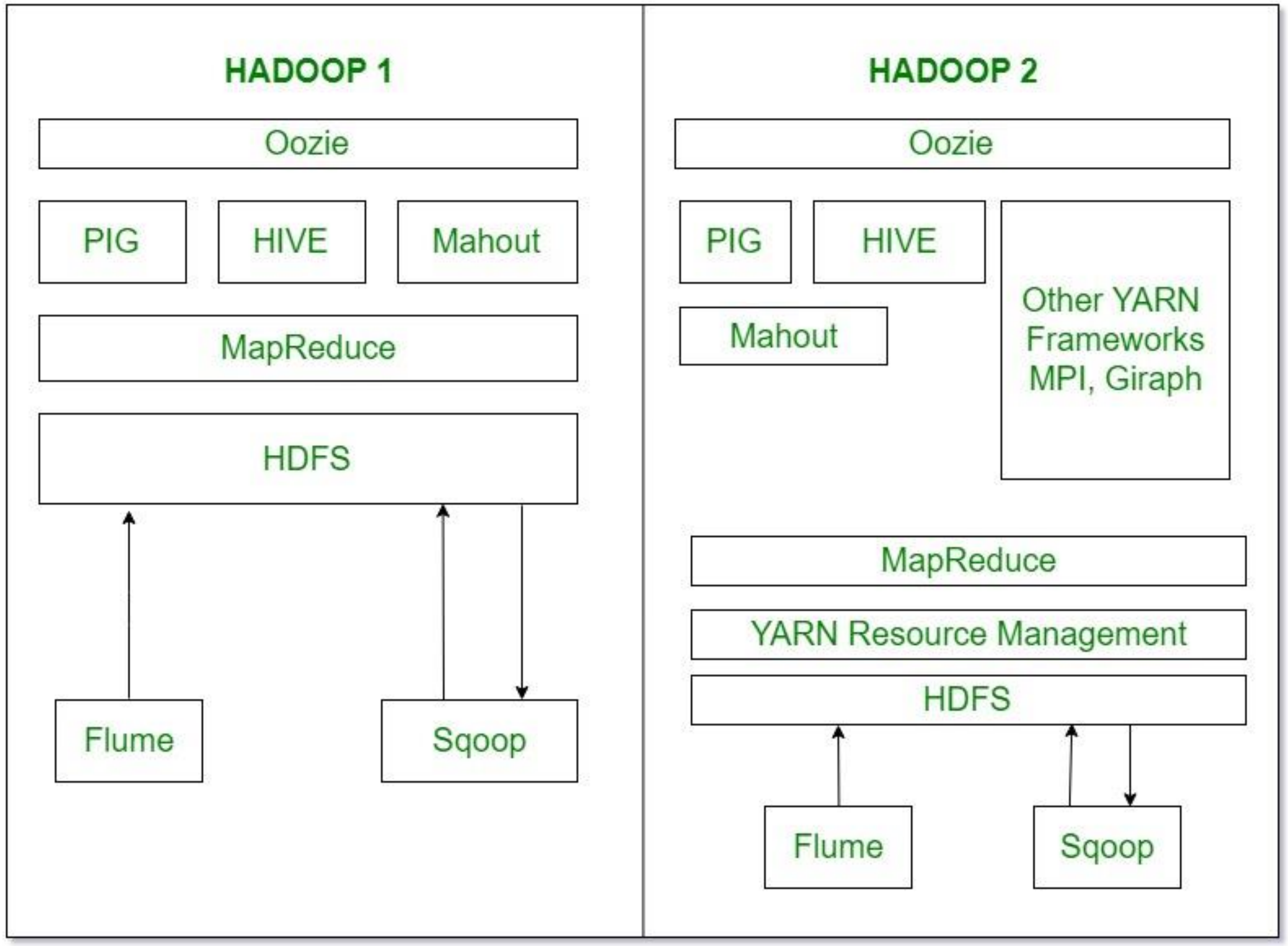
Working

- In *Hadoop 1*, there is HDFS which is used for storage and top of it, Map Reduce which works as Resource Management as well as Data Processing. Due to this workload on Map Reduce, it will affect the performance.
- In *Hadoop 2*, there is again HDFS which is again used for storage and on the top of HDFS, there is YARN which works as Resource Management. It basically allocates the resources and keeps all the things going on.

Limitations

- *Hadoop 1* is a Master-Slave architecture. It consists of a single master and multiple slaves. Suppose if master node got crashed then irrespective of your best slave nodes, your cluster will be destroyed. Again for creating that cluster means copying system files, image files, etc. on another system is too much time consuming which will not be tolerated by organizations in today's time. *Hadoop 2* is also a Master-Slave architecture. But this consists of multiple masters (i.e active namenodes and standby namenodes) and multiple slaves. If here master node got crashed then standby master node will take over it. You can make multiple combinations of active-standby nodes. *Thus Hadoop 2 will eliminate the problem of a single point of failure*.

Ecosystem


HADOOP 1
Oozie
PIG
HIVE
Mahout
MapReduce
HDFS
Flume
Sqoop
HADOOP 2
Oozie
PIG
HIVE
Mahout
Other YARN Frameworks MPI, Giraph
MapReduce
YARN Resource Management
HDFS
Flume
Sqoop

- *Oozie* is basically Work Flow Scheduler. It decides the particular time of jobs to execute according to their dependency.
- *Pig, Hive and Mahout* are data processing tools that are working on the top of Hadoop.
- *Sqoop* is used to import and export structured data. You can directly import and export the data into HDFS using SQL database.
- *Flume* is used to import and export the unstructured data and streaming data

Windows Support

- in **Hadoop 1** there is no support for Microsoft Windows provided by Apache whereas in **Hadoop 2** there is support for Microsoft windows.

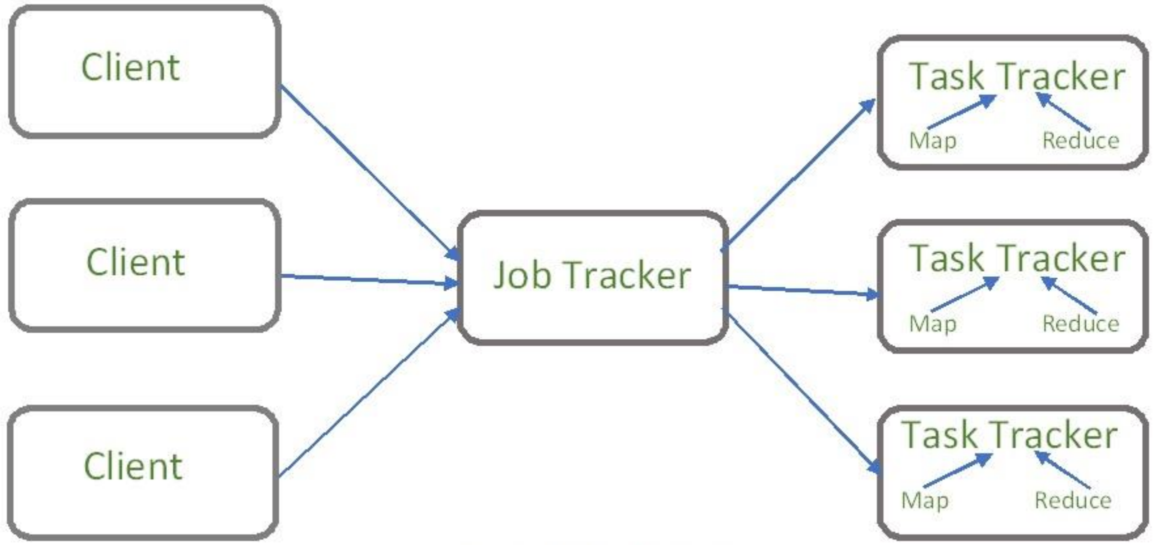
Sr. No.	Key	Hadoop 1	Hadoop 2
1	New Components and API	As Hadoop 1 introduced prior to Hadoop 2 so has some less components and APIs as compare to that of Hadoop 2.	On other hand Hadoop 2 introduced after Hadoop 1 so has more components and APIs as compare to Hadoop 1 such as YARN API,YARN FRAMEWORK, and enhanced Resource Manager.
2	Support	Hadoop 1 only supports MapReduce processing model in its architecture and it does not support non MapReduce tools.	On other hand Hadoop 2 allows to work in MapReducer model as well as other distributed computing models like Spark, Hama, Giraph, Message Passing Interface) MPI & HBase coprocessors.
3	Resource Management	Map reducer in Hadoop 1 is responsible for processing and cluster-resource management.	On other hand in case of Hadoop 2 for cluster resource management YARN is used while processing management is done using different processing models.
4	Scalability	As Hadoop 1 is prior to Hadoop 2 so comparatively less scalable than Hadoop 2 and in context of scaling of nodes it is limited to 4000 nodes per cluster	On other hand Hadoop 2 has better scalability than Hadoop 1 and is scalable up to 10000 nodes per cluster.
5	Implementation	Hadoop 1 is implemented as it follows the concepts of slots which can be used to run a Map task or a Reduce task only.	On other hand Hadoop 2 follows concepts of containers that can be used to run generic tasks.
6	Windows Support	Initially in Hadoop 1 there is no support for Microsoft Windows provided by Apache.	On other hand with an advancement in version of Hadoop Apache provided support for Microsoft windows in Hadoop 2.

Hadoop YARN

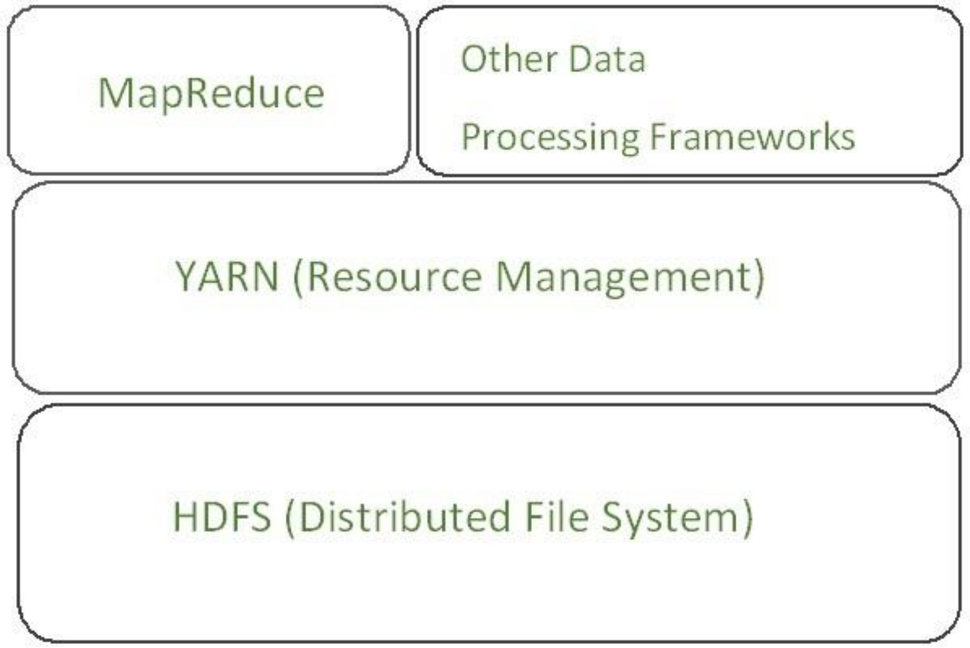
Hadoop YARN

- YARN stands for “***Yet Another Resource Negotiator***“. It was introduced in Hadoop 2.0 to remove the bottleneck on Job Tracker which was present in Hadoop 1.0. YARN was described as a “*Redesigned Resource Manager*” at the time of its launching, but it has now evolved to be known as large-scale distributed operating system used for Big Data processing.

YARN architecture basically separates resource management layer from the processing layer. In Hadoop 1.0 version, the responsibility of Job tracker is split between the resource manager and application manager



Hadoop 1.0 architecture

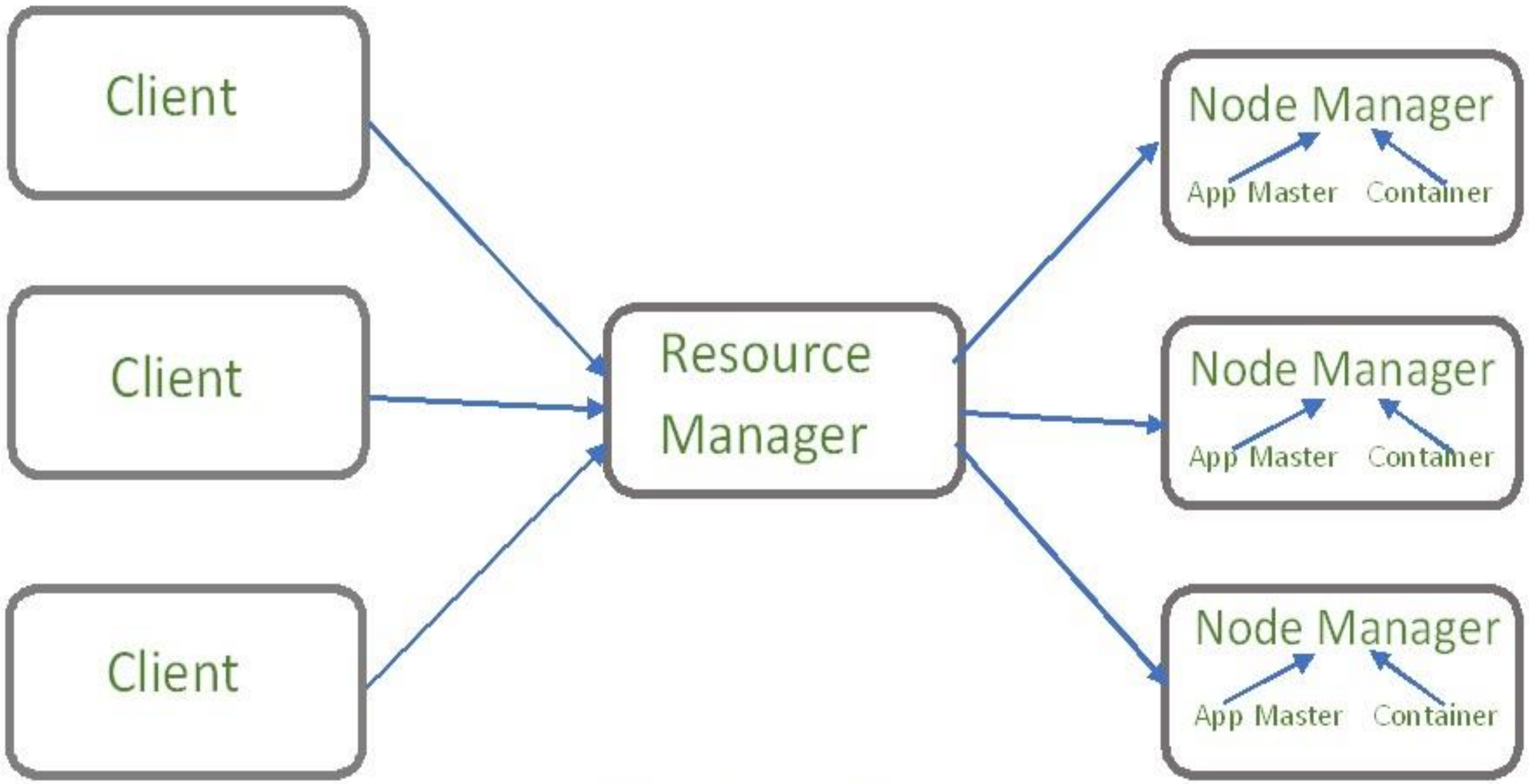

MapReduce
Other Data Processing Frameworks
YARN (Resource Management)
HDFS (Distributed File System)

Hadoop 2.0

- YARN also allows different data processing engines like graph processing, interactive processing, stream processing as well as batch processing to run and process data stored in HDFS (Hadoop Distributed File System) thus making the system much more efficient. Through its various components, it can dynamically allocate various resources and schedule the application processing. For large volume data processing, it is quite necessary to manage the available resources properly so that every application can leverage them.

YARN Features

- **Scalability:** The scheduler in Resource manager of YARN architecture allows Hadoop to extend and manage thousands of nodes and clusters.
- **Compatibility:** YARN supports the existing map-reduce applications without disruptions thus making it compatible with Hadoop 1.0 as well.
- **Cluster Utilization:**Since YARN supports Dynamic utilization of cluster in Hadoop, which enables optimized Cluster Utilization.
- **Multi-tenancy:** It allows multiple engine access thus giving organizations a benefit of multi-tenancy.



Hadoop Yarn architecture

Components of YARN architecture

- **Client**
- **Resource Manager**
- **Node Manager**
- **Application Master**
- **Container**

- **Client:** It submits map-reduce jobs.
- **Resource Manager:** It is the master daemon of YARN and is responsible for resource assignment and management among all the applications. Whenever it receives a processing request, it forwards it to the corresponding node manager and allocates resources for the completion of the request accordingly. It has two major components:
    - **Scheduler:** It performs scheduling based on the allocated application and available resources. It is a pure scheduler, means it does not perform other tasks such as monitoring or tracking and does not guarantee a restart if a task fails. The YARN scheduler supports plugins such as Capacity Scheduler and Fair Scheduler to partition the cluster resources.
    - **Application manager:** It is responsible for accepting the application and negotiating the first container from the resource manager. It also restarts the Application Master container if a task fails.

Node Manager

- It take care of individual node on Hadoop cluster and manages application and workflow and that particular node. Its primary job is to keep-up with the Resource Manager. It registers with the Resource Manager and sends heartbeats with the health status of the node. It monitors resource usage, performs log management and also kills a container based on directions from the resource manager. It is also responsible for creating the container process and start it on the request of Application master.

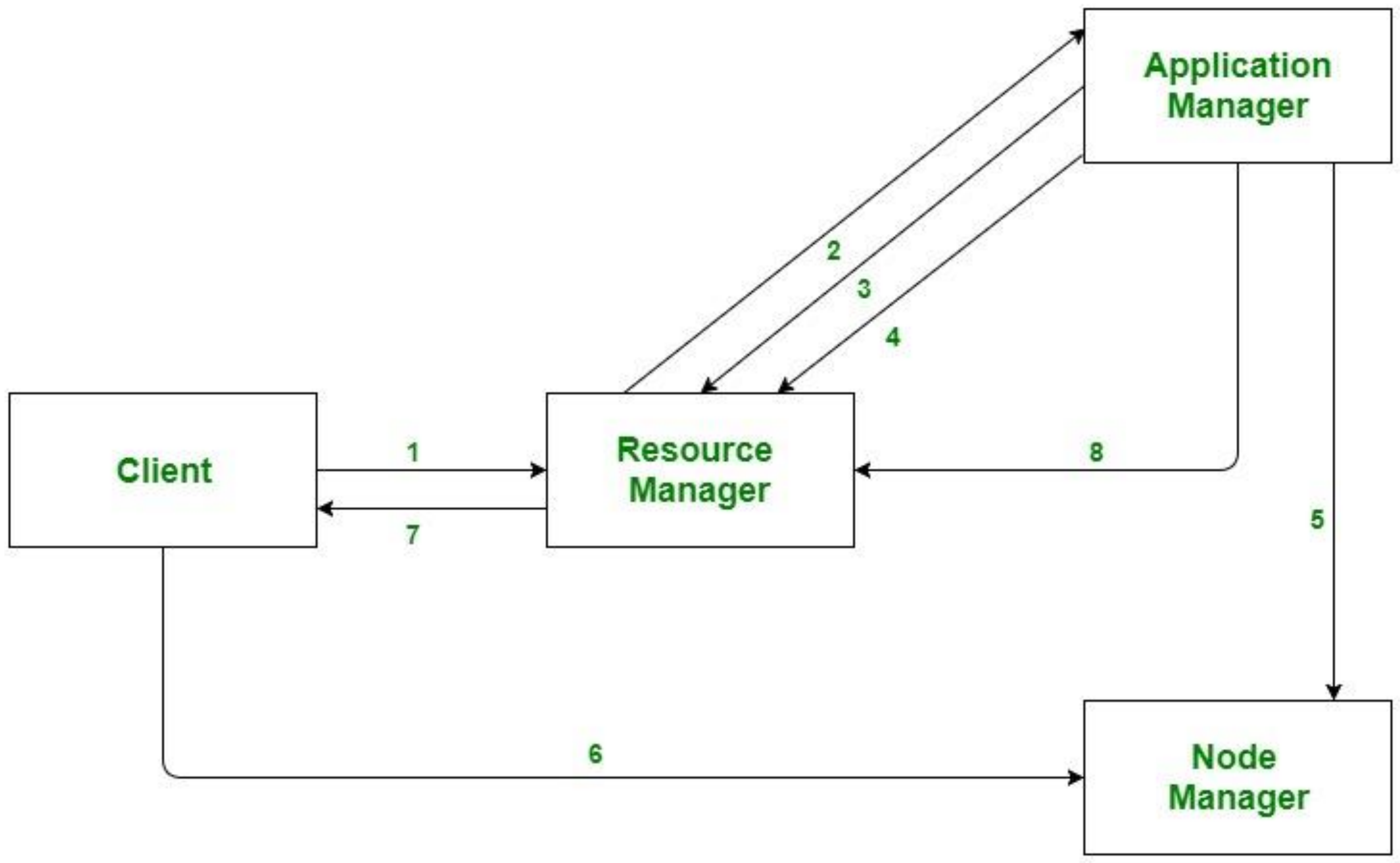
Application Master

- An application is a single job submitted to a framework. The application master is responsible for negotiating resources with the resource manager, tracking the status and monitoring progress of a single application. The application master requests the container from the node manager by sending a Container Launch Context(CLC) which includes everything an application needs to run. Once the application is started, it sends the health report to the resource manager from time-to-time.

Container

- It is a collection of physical resources such as RAM, CPU cores and disk on a single node. The containers are invoked by Container Launch Context(CLC) which is a record that contains information such as environment variables, security tokens, dependencies etc.

Application workflow in Hadoop YARN



1. Client submits an application
2. The Resource Manager allocates a container to start the Application Manager
3. The Application Manager registers itself with the Resource Manager
4. The Application Manager negotiates containers from the Resource Manager
5. The Application Manager notifies the Node Manager to launch containers
6. Application code is executed in the container
7. Client contacts Resource Manager/Application Manager to monitor application’s status
8. Once the processing is complete, the Application Manager un-registers with the Resource Manager

Advantages :

- **Flexibility:** YARN offers flexibility to run various types of distributed processing systems such as Apache Spark, Apache Flink, Apache Storm, and others. It allows multiple processing engines to run simultaneously on a single Hadoop cluster.
- **Resource Management:** YARN provides an efficient way of managing resources in the Hadoop cluster. It allows administrators to allocate and monitor the resources required by each application in a cluster, such as CPU, memory, and disk space.
- **Scalability:** YARN is designed to be highly scalable and can handle thousands of nodes in a cluster. It can scale up or down based on the requirements of the applications running on the cluster.
- **Improved Performance:** YARN offers better performance by providing a centralized resource management system. It ensures that the resources are optimally utilized, and applications are efficiently scheduled on the available resources.
- **Security:** YARN provides robust security features such as Kerberos authentication, Secure Shell (SSH) access, and secure data transmission. It ensures that the data stored and processed on the Hadoop cluster is secure.

Disadvantages

- **Complexity:** YARN adds complexity to the Hadoop ecosystem. It requires additional configurations and settings, which can be difficult for users who are not familiar with YARN.
- **Overhead:** YARN introduces additional overhead, which can slow down the performance of the Hadoop cluster. This overhead is required for managing resources and scheduling applications.
- **Latency:** YARN introduces additional latency in the Hadoop ecosystem. This latency can be caused by resource allocation, application scheduling, and communication between components.
- **Single Point of Failure:** YARN can be a single point of failure in the Hadoop cluster. If YARN fails, it can cause the entire cluster to go down. To avoid this, administrators need to set up a backup YARN instance for high availability.
- **Limited Support:** YARN has limited support for non-Java programming languages. Although it supports multiple processing engines, some engines have limited language support, which can limit the usability of YARN in certain environments.