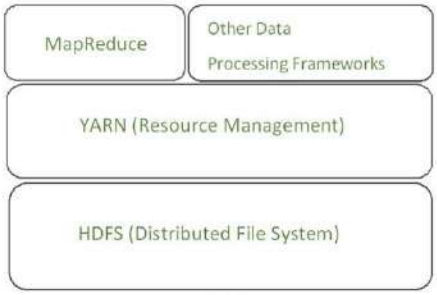
# 1. What is YARN?

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. It has now evolved to be known as a **large-scale distributed operating system used for Big Data processing**.

YARN architecture **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.



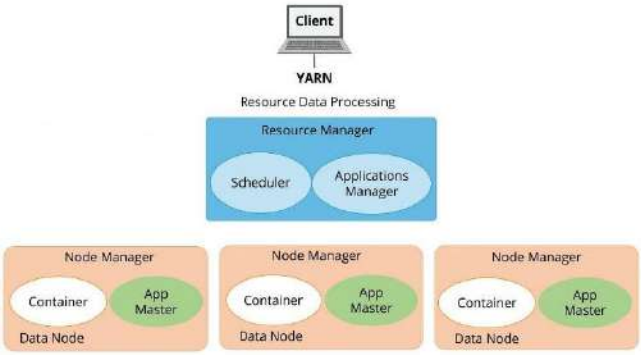
Hadoop 2.0

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.

The components of YARN architecture include:

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

YARN Architecture are shown in the given below diagram.



**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.

## 2. List the functions of YARN.

YARN (Yet Another Resource Negotiator) is the resource management layer in Hadoop. Its primary function is to manage resources in a Hadoop cluster and schedule jobs efficiently. Here are its key functions:

1. **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.

2. **Job Scheduling**: It schedules and prioritizes jobs submitted to the cluster based on various criteria such as resource requirements, job priority, and availability of resources.

3. **Fault Tolerance:** YARN monitors the status of individual nodes and tasks in the cluster. In case of node failures or task failures, it reallocates resources and reschedules tasks to ensure job completion and fault tolerance.

4. **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.

5. **Multi-Tenancy Support:** It supports multi-tenancy by allowing multiple users or applications to share cluster resources securely. YARN provides isolation between different applications running on the same cluster.

6. **Pluggable Scheduler:** YARN supports pluggable scheduling algorithms, allowing administrators to choose the most suitable scheduler for their workload and cluster environment. Schedulers like Capacity Scheduler and Fair Scheduler are commonly used with YARN.

7. **Integration with Hadoop Ecosystem:** YARN integrates seamlessly with other components of the Hadoop ecosystem, such as HDFS (Hadoop Distributed File System), MapReduce, Apache Spark, Apache Hive, Apache HBase, and others, enabling a wide range of data processing and analytics tasks.

These functions collectively make YARN a crucial component of the Hadoop ecosystem, enabling efficient resource management and job execution in large-scale distributed computing environments.

## 3. How does MapReducer along with the YARN resources manager enable faster processing of an application.

MapReduce, along with YARN (Yet Another Resource Negotiator), is a powerful framework in the Hadoop ecosystem designed for processing large datasets in parallel across a distributed cluster of commodity hardware. Here's how they work together to enable faster processing of an application:

1. **Parallel Processing:** MapReduce divides the processing task into smaller sub-tasks called "maps" and "reduces", which can be executed in parallel across multiple nodes in the cluster. YARN enables efficient utilization of the available resources and accelerates the processing speed.

2. **Fault Tolerance:** MapReduce and YARN provide fault tolerance mechanisms. If a node fails during processing, YARN can redistribute the tasks to other healthy nodes in the cluster, ensuring that the computation continues without interruption. This resilience to failures minimizes downtime and ensures high availability of the application.

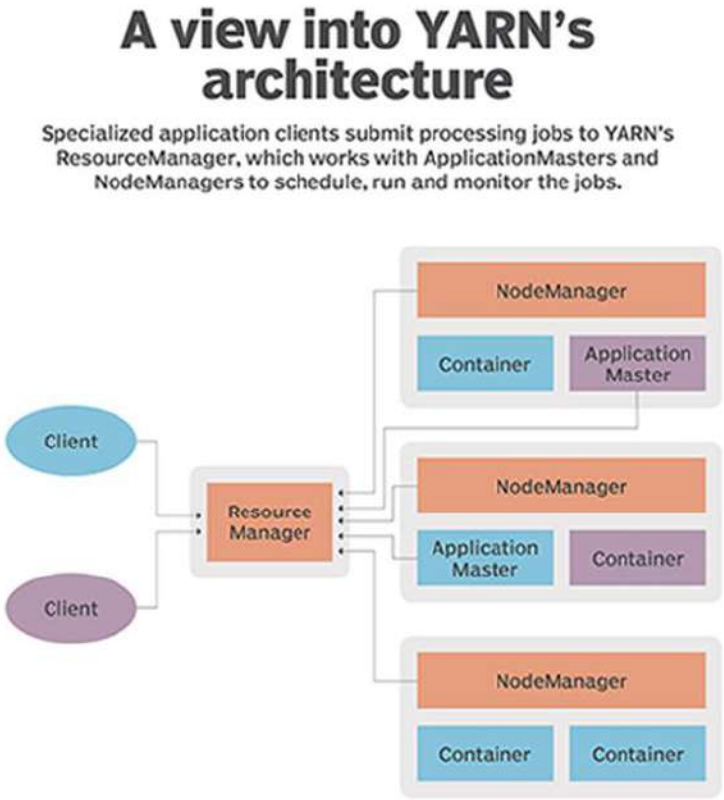
3. **Resource Management:** YARN acts as a resource manager in the Hadoop cluster, dynamically allocating resources (CPU, memory, etc.) to different applications based on their requirements. It efficiently manages and schedules resources among various concurrent applications, preventing resource contention and optimizing resource utilization. This ensures that MapReduce jobs receive the necessary resources to execute efficiently, contributing to faster processing.

4. **Scalability:** Both MapReduce and YARN are designed to scale horizontally, meaning additional nodes can be added to the cluster to handle larger datasets or increased processing demands. This scalability ensures that the system can grow to accommodate the growing workload, maintaining performance levels as the data volume or application complexity increases.

5. **Resource Isolation:** YARN provides resource isolation, ensuring that the resources allocated to one application do not interfere with the resources allocated to another. This prevents resource hogging by any single application, leading to more predictable performance and faster processing times for all applications running on the cluster.

Overall, the combination of MapReduce and YARN enables faster processing of applications by leveraging parallelism, fault tolerance, efficient resource management, data locality optimization, scalability, and resource isolation within a distributed computing environment.

## 4. Explain the YARN architecture.




The components of YARN architecture include:

**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.

## 5. What do you understand by Resource Manager?

**Follow Question 4**

## 6. How can you access logs in Hadoop2 using YARN commands?

In Hadoop 2, YARN (Yet Another Resource Negotiator) is responsible for resource management and job scheduling. To access logs for applications running on YARN, you typically use the yarn logs command. Here's how you can do it:

**Identify the Application ID:** Each application running on YARN is assigned a unique Application ID. You need to know the Application ID of the specific application whose logs you want to access.

**Use the yarn logs Command:** Once you have the Application ID, you can use the yarn logs command followed by the Application ID to access the logs. Here's the basic syntax:

yarn logs -applicationId <Application ID>

**Specify the Container ID (Optional):** If your application has multiple containers, you can specify a particular container ID to fetch the logs for that specific container. The container ID can be obtained from the application's details.

yarn logs -applicationId <Application ID> -containerId <Container ID>

**Viewing Logs on the Command Line:** Running the above command will fetch the logs and display them on the command line.

**Redirecting Logs to a File (Optional):** If you want to save the logs to a file, you can redirect the output to a file using standard shell redirection operators.

yarn logs -applicationId <Application ID> logs.txt

By following these steps, you can access and view logs for applications running on YARN in Hadoop 2.

## 7. Discuss the advantages of YARN over MapReduce

YARN (Yet Another Resource Negotiator) is a significant advancement over the traditional MapReduce framework in Hadoop. Here are some advantages of YARN over MapReduce:

1. **Improved Resource Management:** YARN decouples the resource management and job scheduling functions of Hadoop, allowing for more efficient resource utilization. Unlike MapReduce, which had a fixed resource allocation model, YARN dynamically allocates resources to applications based on their needs, leading to better cluster utilization.

2. **Support for Multiple Processing Models:** While MapReduce is primarily designed for batch processing, YARN supports multiple processing models, including interactive querying (Apache Tez), real-time processing (Apache Storm), and graph processing (Apache Giraph), among others. This flexibility enables Hadoop to address a wider range of use cases beyond batch processing.

3. **Enhanced Scalability:** YARN provides better scalability compared to MapReduce by allowing multiple applications to run concurrently on the same cluster. It efficiently manages cluster resources and can handle a larger number of applications simultaneously, improving overall cluster throughput.

4. **Faster Job Execution**: YARN enables faster job execution by supporting in-memory processing frameworks like Apache Tez and Apache Spark. These frameworks leverage data locality and optimized task execution strategies, resulting in reduced job completion times compared to traditional MapReduce jobs.

Overall, YARN's architectural improvements provide better resource utilization, scalability, and support for diverse processing models compared to the traditional MapReduce framework, making it a more versatile and efficient resource management solution for big data processing in Hadoop.

# 8. Explain in detail how Job schedules in Hadoop.

***Hadoop – Schedulers and Types of Schedulers***

In Hadoop, we can receive multiple jobs from different clients to perform. In **Hadoop 1,** Map-Reduce Framework is responsible for scheduling and monitoring the tasks given by different clients in a Hadoop cluster.

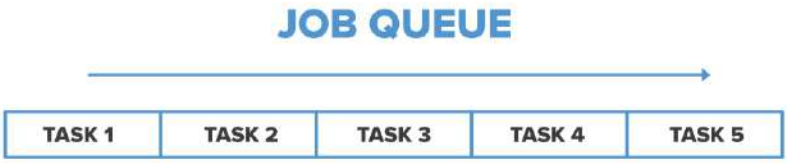
But in Hadoop 2, the Scheduler, major component of Resource Manager in YARN is dedicated to scheduling the jobs.

**There are mainly 3 types of Schedulers in Hadoop:**

- FIFO (First In First Out) Scheduler.
- Capacity Scheduler.
- Fair Scheduler.

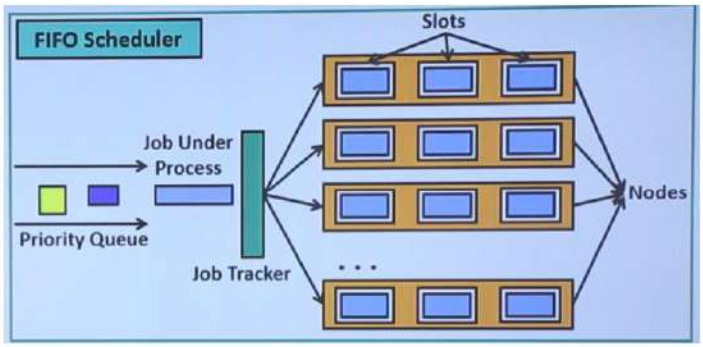
These Schedulers are actually a kind of algorithm that we use to schedule tasks in a Hadoop cluster when we receive requests from different-different clients.

A **Job queue** is nothing but the collection of various tasks that we have received from our various clients. The tasks are available in the queue and we need to schedule this task on the basis of our requirements.




**1. FIFO Scheduler**

As the name suggests FIFO i.e. First In First Out, so the tasks or applications that come first will be served first. This is the default Scheduler of Hadoop 1. The tasks are placed in a queue and the **tasks are performed based on their submission order**. In this method, once the job is scheduled, **no intervention is allowed**. So sometimes the high-priority process has to wait for a long time since the priority of the task does not matter in this method.




**Advantage:**

- No need for configuration
- First Come First Serve

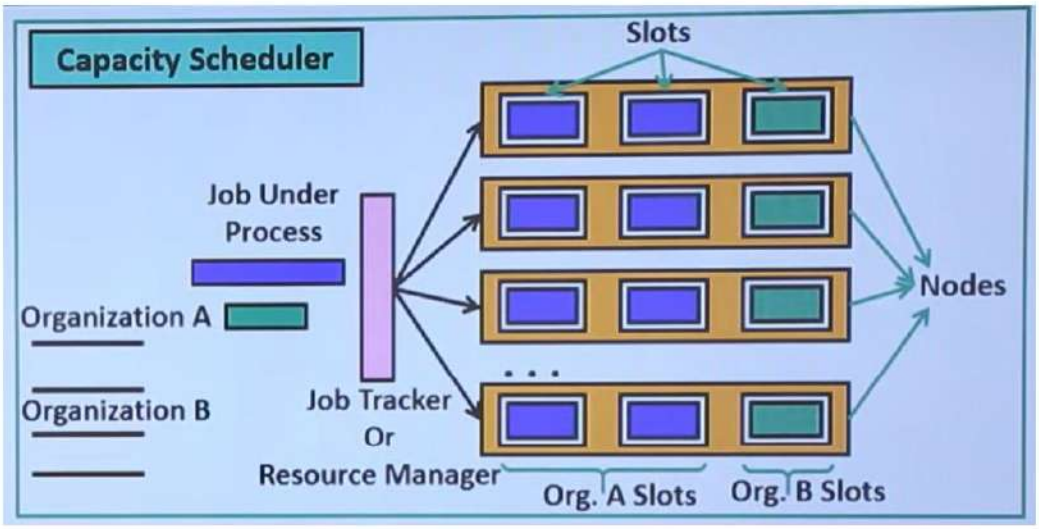
- simple to execute

**Disadvantage:**

- Priority of task doesn't matter, so high priority jobs need to wait
- Not suitable for shared cluster

### 2. Capacity Scheduler

In Capacity Scheduler we have multiple job queues for scheduling our tasks. In Capacity Scheduler corresponding for each job queue, we provide some slots or cluster resources for performing job operation. Each job queue has it's own slots to perform its task. In case we have tasks to perform in only one queue then the tasks of that queue can access the slots of other queues also as they are free to use, and when the new task enters to some other queue then jobs in running in its own slots of the cluster are replaced with its own job.




**Advantage:**

- Best for working with Multiple clients or priority jobs in a Hadoop cluster
- Maximizes throughput in the Hadoop cluster

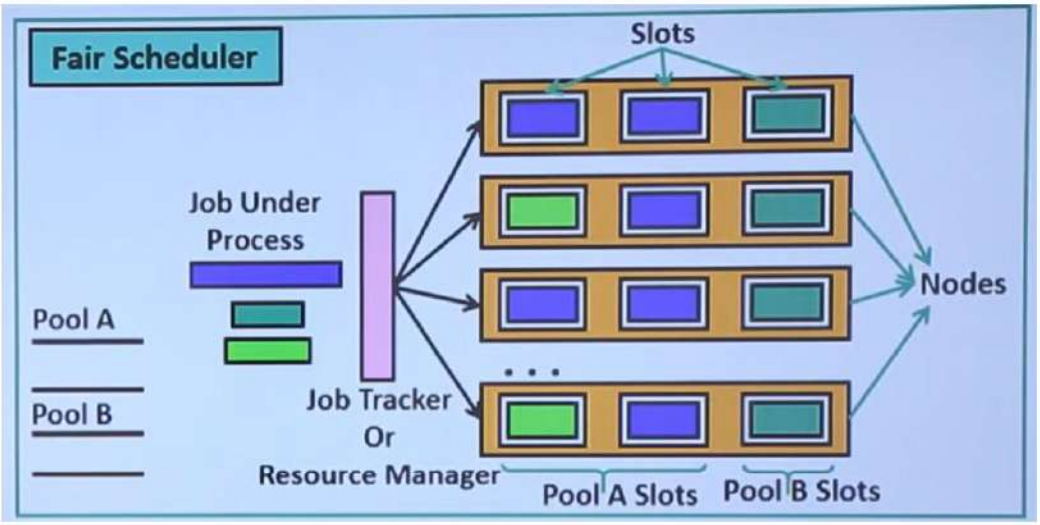
**Disadvantage:**

- More complex
- Not easy to configure for everyone

### 3. Fair Scheduler

The Fair Scheduler is very much similar to that of the capacity scheduler. The priority of the job is kept in consideration. With the help of Fair Scheduler, the YARN applications can share the resources in the large Hadoop Cluster and these resources are maintained dynamically so no need for prior capacity.

In Fair Scheduler whenever any high priority job arises in the same queue, the task is processed in parallel by replacing some portion from the already dedicated slots.




**Advantages:**

- Resources assigned to each application depend upon its priority.
- it can limit the concurrent running task in a particular pool or queue.

**Disadvantages:** The configuration is required.

## 9. Discuss fair scheduling in YARN.

## 10. Discuss Capacity scheduling in YARN.

**Follow Question 8**