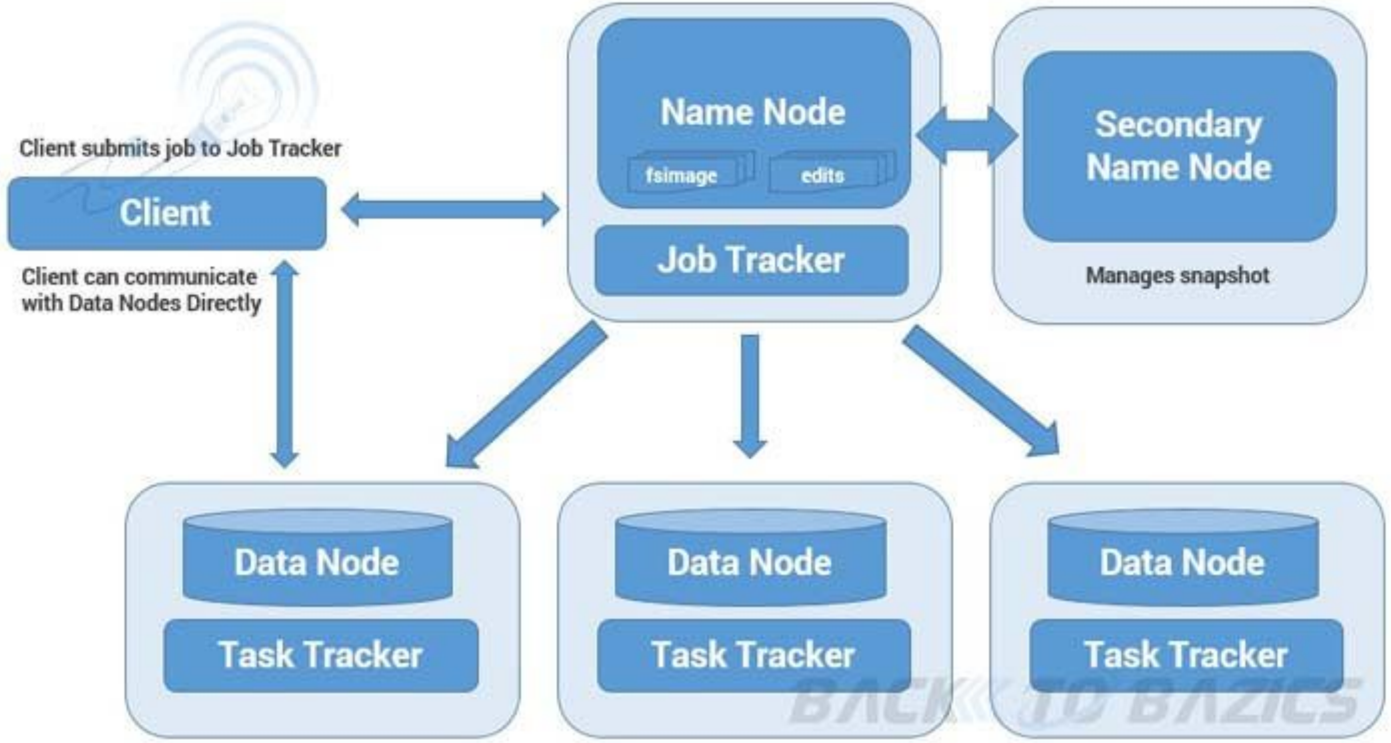
Hadoop


Client submits job to Job Tracker
Client
Client can communicate with Data Nodes Directly
Name Node
fsimage
edits
Job Tracker
Secondary Name Node
Manages snapshot
Data Node
Task Tracker
Data Node
Task Tracker
Data Node
Task Tracker
BACK TO BAZICS

HDFS in Hadoop 1.x mainly has 3 daemons which are Name Node, Secondary Name Node and Data Node.

Name Node

- There is only single instance of this process runs on a cluster and that is on a master node
- It is responsible for manage metadata about files distributed across the cluster
- It manages information like location of file blocks across cluster and it's permission
- This process reads all the metadata from a file named `fsimage` and keeps it in memory
- After this process is started, it updates metadata for newly added or removed files in RAM
- It periodically writes the changes in one file called `edits` as edit logs
- This process is a heart of HDFS, if it is down HDFS is not accessible any more

Secondary Name Node

- For this also, only single instance of this process runs on a cluster
- This process can run on a master node (for smaller clusters) or can run on a separate node (in larger clusters) depends on the size of the cluster
- One misinterpretation from name is *"This is a backup Name Node"* but **IT IS NOT!!!!!**
- It manages the metadata for the Name Node. In the sense, it reads the information written in edit logs (by Name Node) and creates an updated file of current cluster metadata
- Than it transfers that file back to Name Node so that `fsimage` file can be updated
- So, whenever Name Node daemon is restarted it can always find updated information in `fsimage` file

Data Node

- There are many instances of this process running on various slave nodes(referred as Data nodes)
- It is responsible for storing the individual file blocks on the slave nodes in Hadoop cluster
- Based on the replication factor, a single block is replicated in multiple slave nodes(only if replication factor is > 1) to prevent the data loss
- Whenever required, this process handles the access to a data block by communicating with Name Node
- This process periodically sends heart bits to Name Node to make Name Node aware that slave process is running

Job Tracker

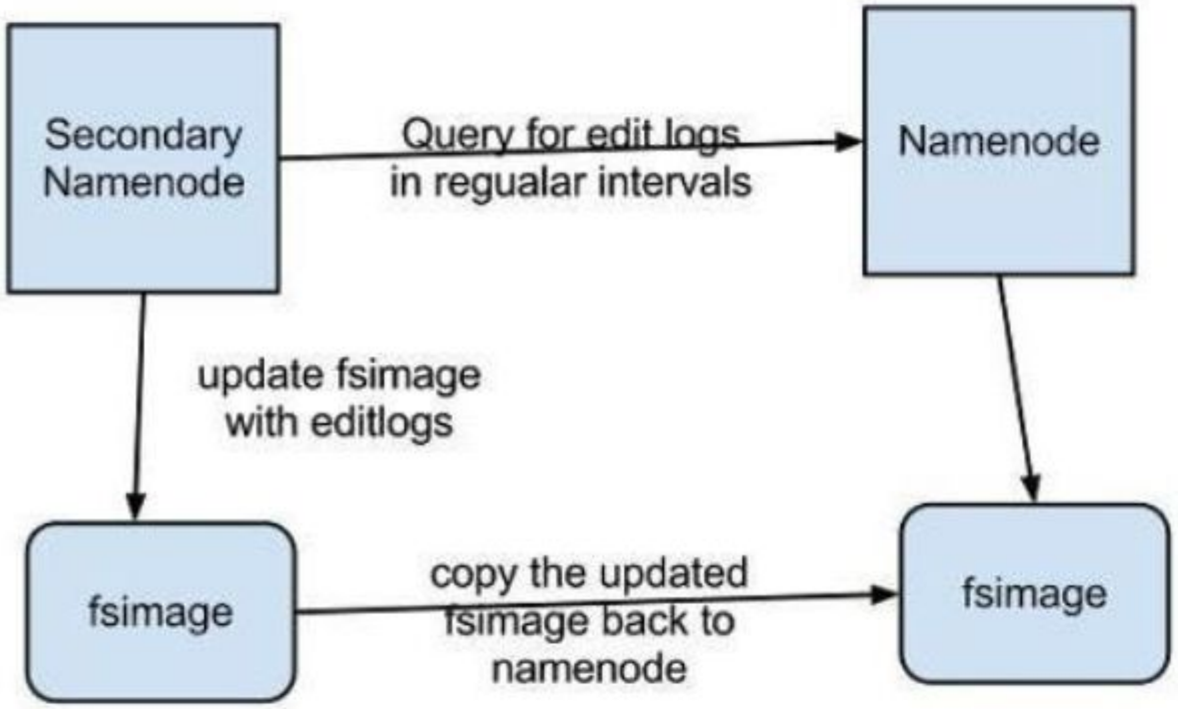
- Only one instance of this process runs on a master node same as Name Node
- Any MapReduce job is submitted to Job Tracker first
- Job Tracker checks for the location various file blocks used in MapReduce processing
- Than it initiates the separate tasks on various Data Nodes(where blocks are present) by communicating with Task Tracker Daemons

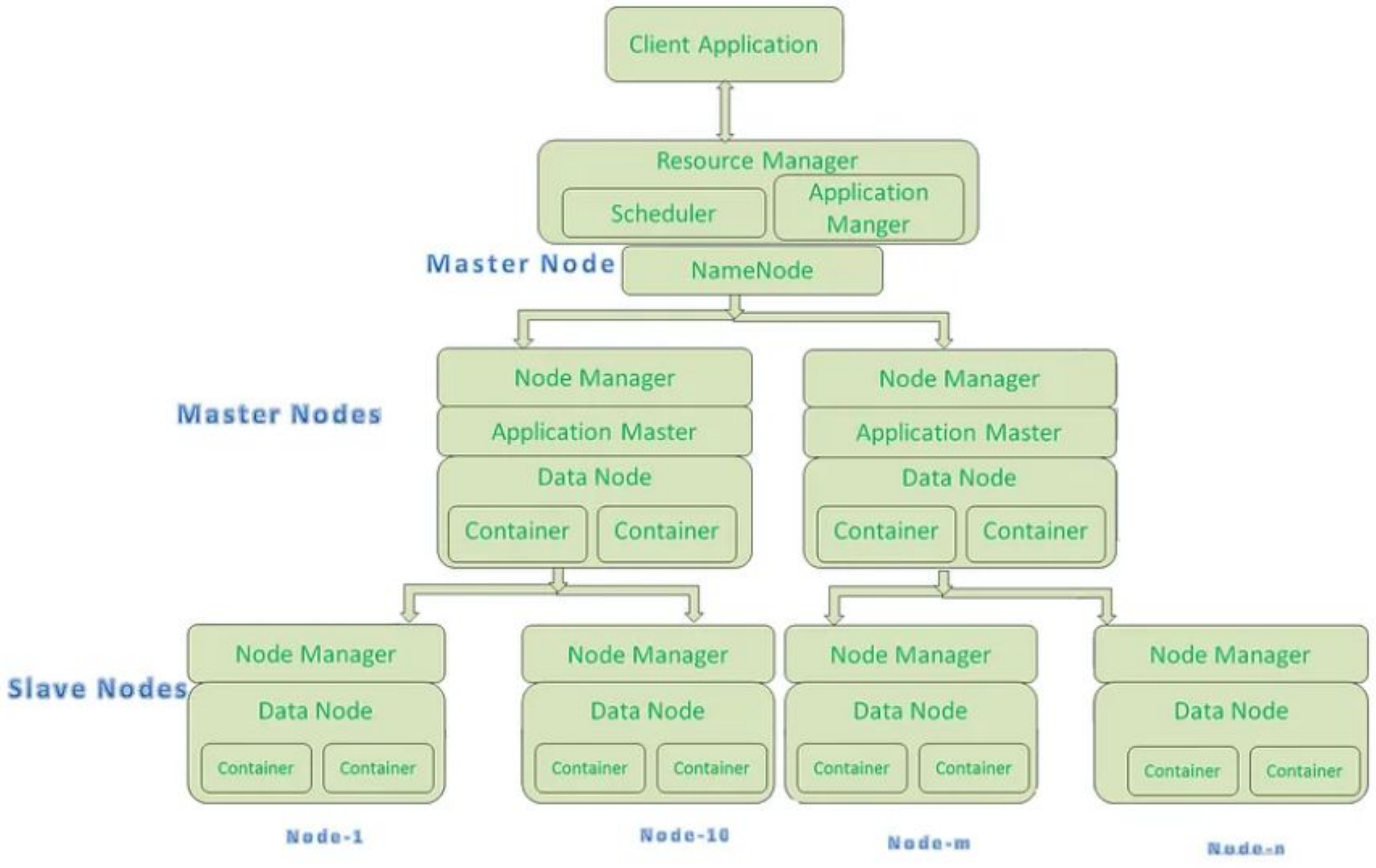
Task Tracker

- This process has multiple instances running on the slave nodes(typically runs on the slave nodes where Data Node process is running)
- It receives the job information from Job Tracker daemon and initiates a task on that slave node
- In most of the cases, Task Tracker initiates the task on the same node where there physical data block is present
- Same as Data Node daemon, this process also periodically sends heart bits to Job Tracker to make Job Tracker aware that slave process is running

Limitations of Hadoop 1.x Architecture

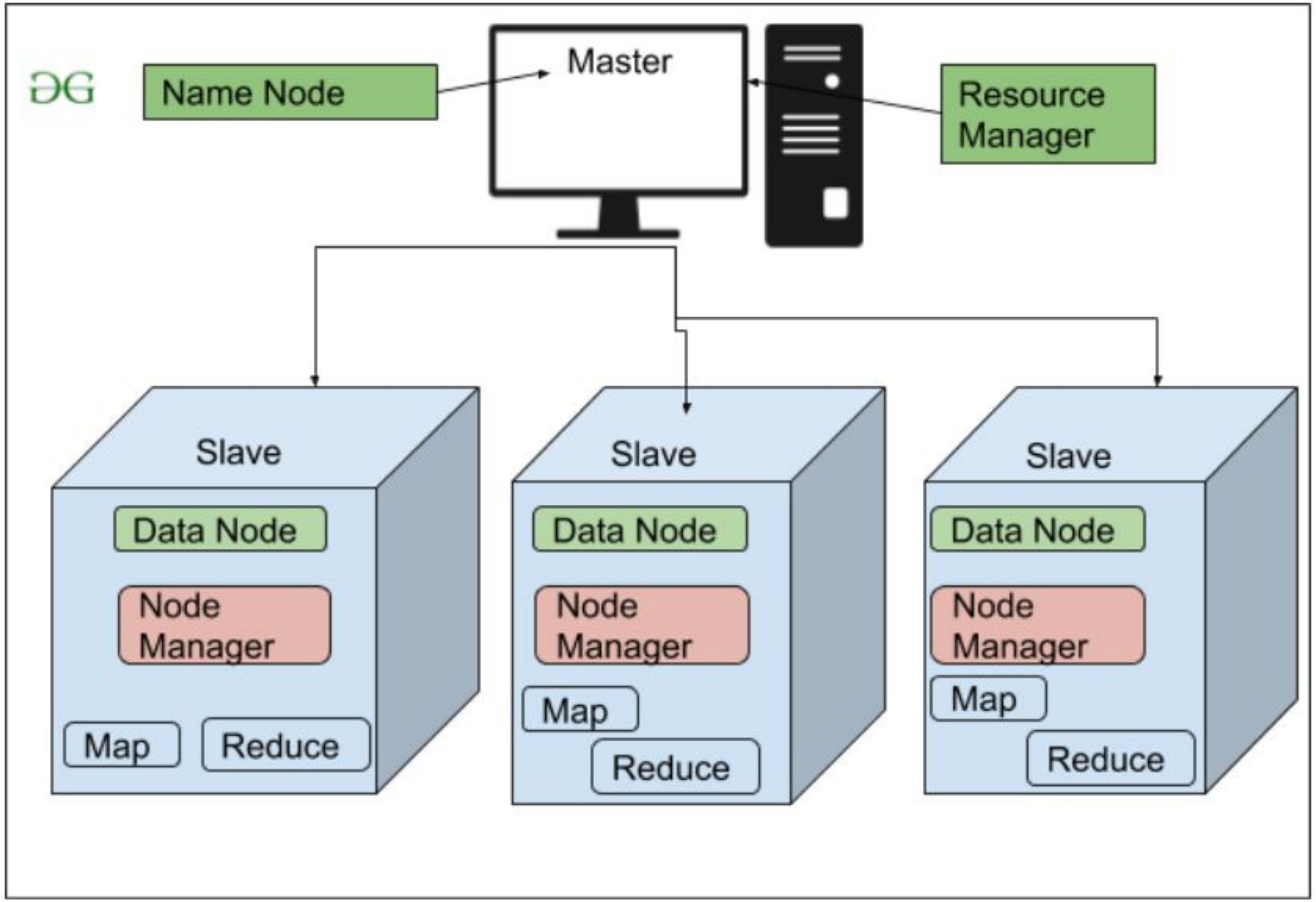
- As we also have seen in above description, major drawback of Hadoop 1.x Architecture is Single Point of Failure as there is no backup Name Node.
- Job scheduling, resource management and job monitoring are being done by Job Tracker which is tightly coupled with Hadoop. So Job Tracker is not able to manage resources outside Hadoop.
- Job Tracker has to coordinate with all task tracker so in a very big cluster it will be difficult to manage huge number of task trackers altogether.
- Due to single Name Node, there is no concept of namespaces in Hadoop 1.x. So everything is being managed under single namespace.
- Using Hadoop 1.x architecture, Hadoop Cluster can be scaled upto ~4000 nodes. Scalablity beyond that may cause performance degradation and increasing task failure ratio.

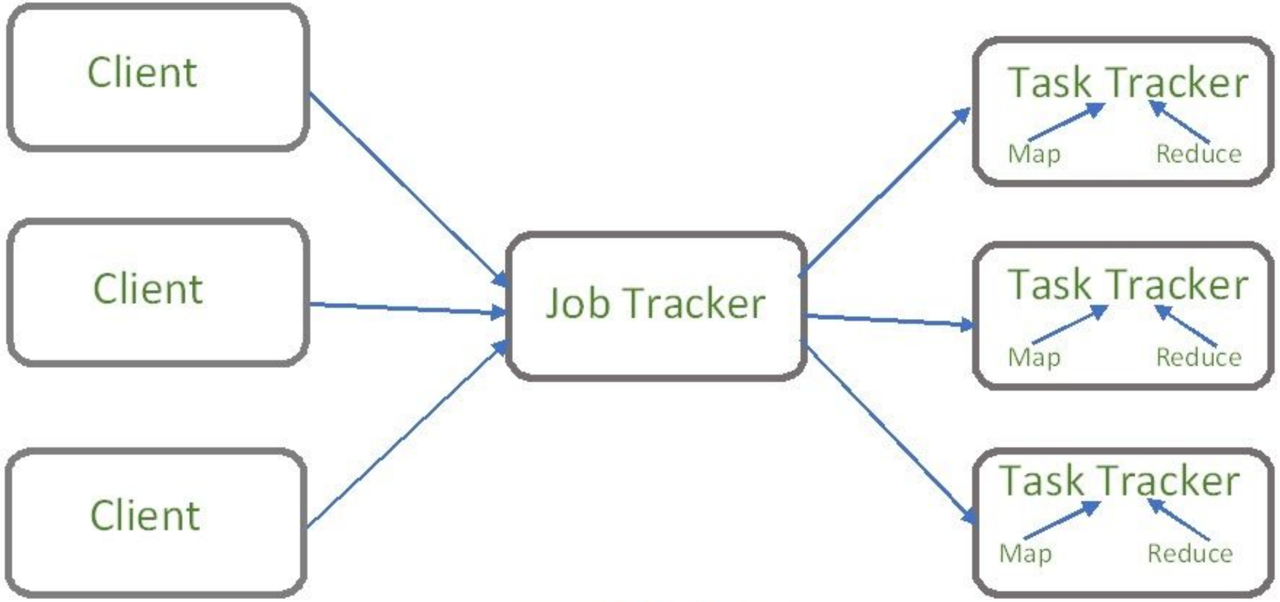

Secondary Namenode
Query for edit logs in regualar intervals
Namenode
update fsimage with editlogs
fsimage
copy the updated fsimage back to namenode
fsimage



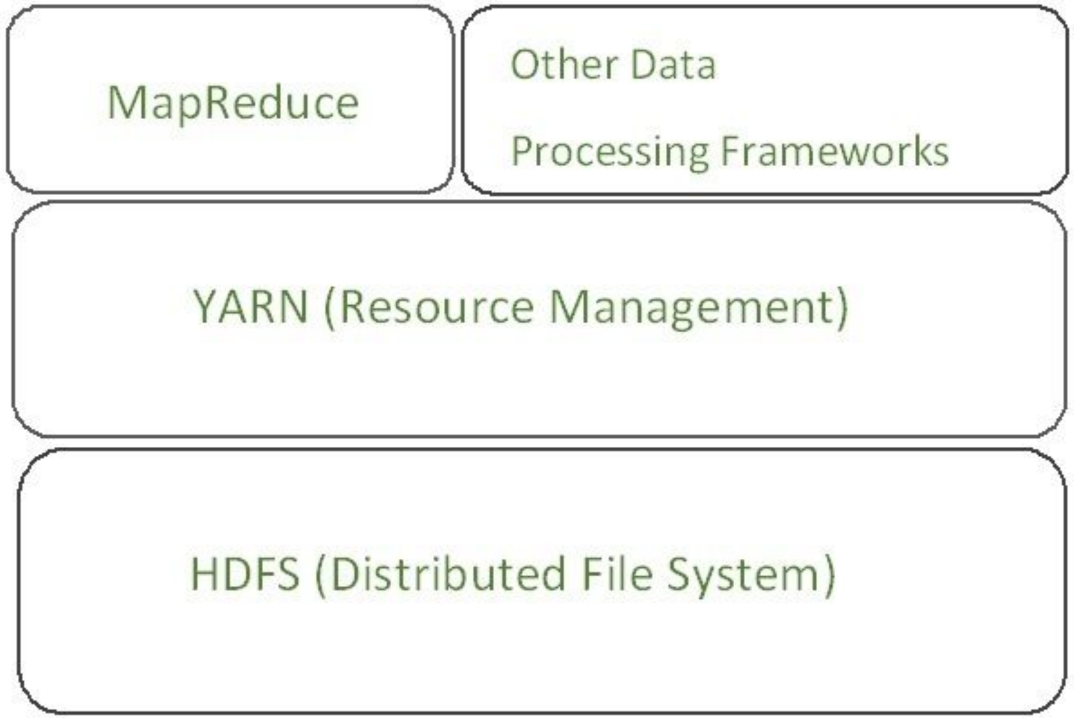
Hadoop 2.x In-Detail Architecture

Hadoop 2.x (HDFS and YARN features)

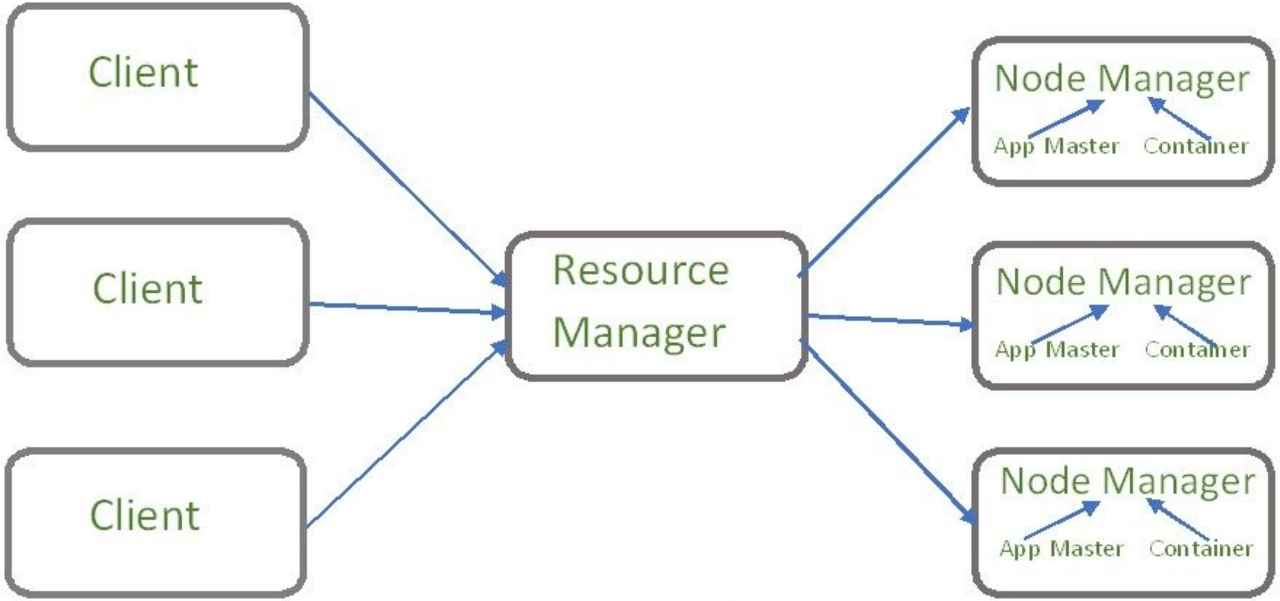

Name Node
Master
Resource Manager
Slave
Data Node
Node Manager
Map
Reduce
Slave
Data Node
Node Manager
Map
Reduce
Slave
Data Node
Node Manager
Map
Reduce



Hadoop 1.0 architecture


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

Hadoop 2.0



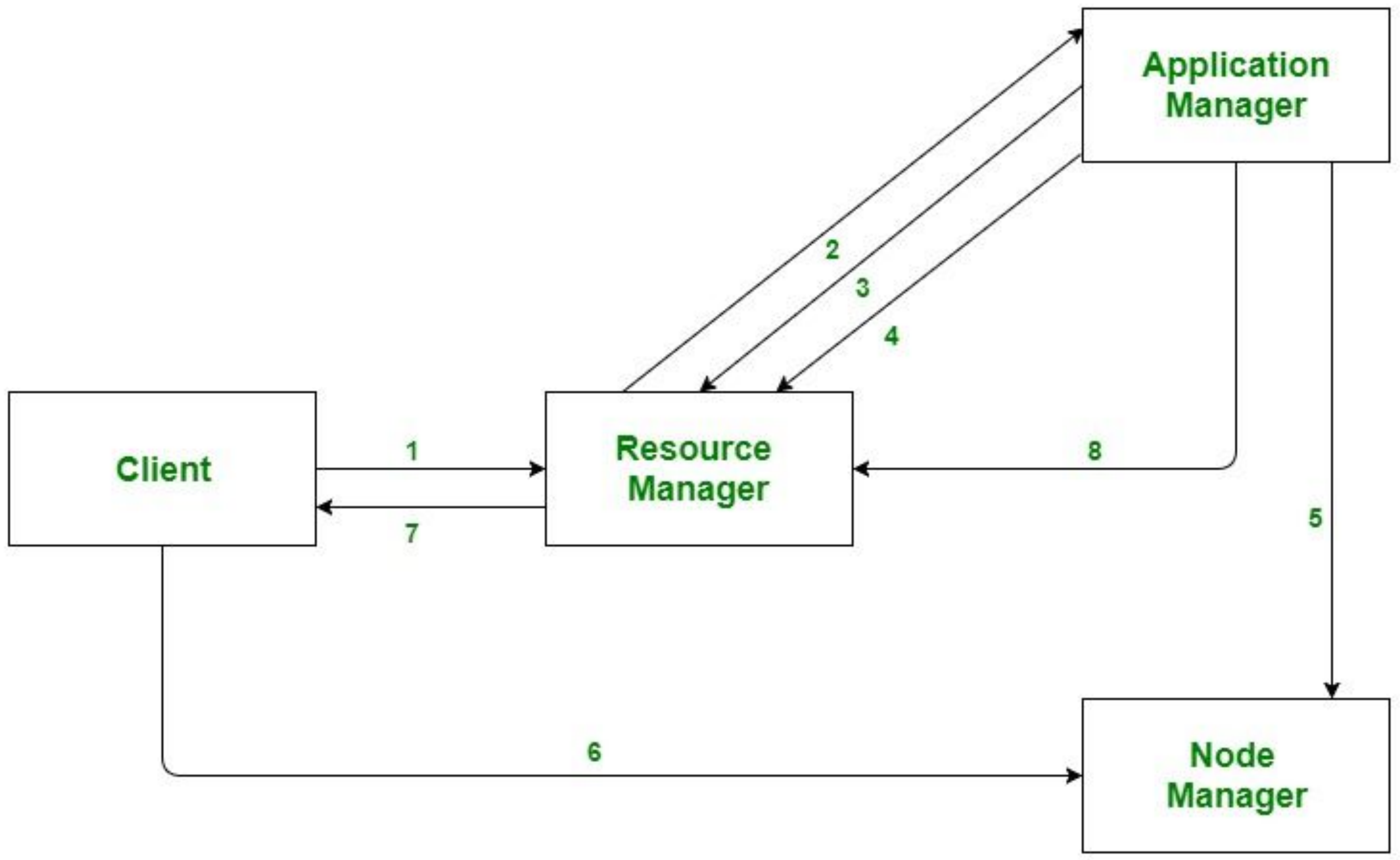
Hadoop Yarn architecture

The main 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.

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.