List the components of Hadoop 2.x and explain them in detail.

The 3 main components of Hadoop 2.x are:

1. HDFS

2. MapReduce

3. YARN

**1.** **HDFS:** The Hadoop Distributed File System (HDFS) is the primary storage system used by [Hadoop](http://searchcloudcomputing.techtarget.com/definition/Hadoop) applications.

HDFS is a [distributed file system](http://searchcio-midmarket.techtarget.com/definition/distributed-file-system) that provides high-performance access to [data](http://searchdatamanagement.techtarget.com/definition/data) across Hadoop clusters. Like other Hadoop-related technologies, HDFS has become a key tool for managing pools of [big data](http://searchcloudcomputing.techtarget.com/definition/big-data-Big-Data) and supporting [big data analytics](http://searchbusinessanalytics.techtarget.com/definition/big-data-analytics) applications.

Because HDFS typically is deployed on low-cost commodity [hardware](http://searchcio-midmarket.techtarget.com/definition/hardware), server failures are common. The file system is designed to be highly [fault-tolerant](http://searchcio-midmarket.techtarget.com/definition/fault-tolerant), however, by facilitating the rapid transfer of data between compute nodes and enabling Hadoop systems to continue running if a [node](http://searchnetworking.techtarget.com/definition/node) fails. That decreases the risk of [catastrophic failure](http://searchwindowsserver.techtarget.com/definition/catastrophic-failure), even in the event that numerous nodes fail.

When HDFS takes in data, it breaks the information down into separate pieces and distributes them to different nodes in a cluster, allowing for [parallel processing](http://whatis.techtarget.com/definition/parallel-processing-software). The file system also copies each piece of data multiple times and distributes the copies to individual nodes, placing at least one copy on a different [server rack](http://whatis.techtarget.com/definition/rack-server-rack-mounted-server) than the others. As a result, the data on nodes that crash can be found elsewhere within a cluster, which allows processing to continue while the failure is resolved.

HDFS is built to support applications with large data sets, including individual files that reach into the terabytes. It uses a master/slave architecture, with each cluster consisting of a single NameNode that manages file system operations and supporting DataNodes that manage data storage on individual compute nodes.

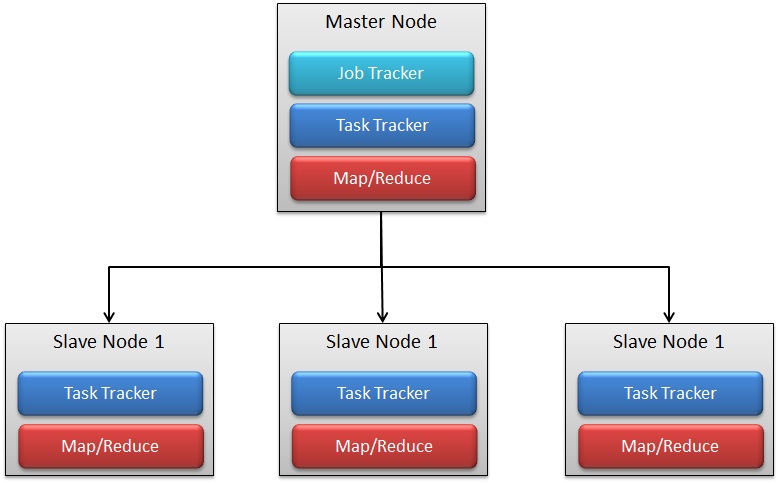


1. **MapReduce:** MapReduce is a programming model which is used to process large data sets in a batch processing manner.   
   A MapReduce program is composed of a Map() procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name)and a Reduce() procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies).

MapReduce is composed of mainly two components.

1. Job tracker

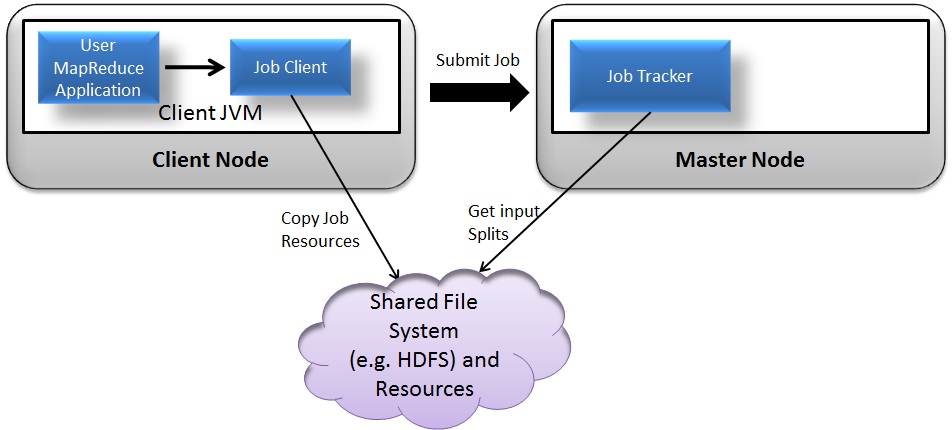
2. Task tracker

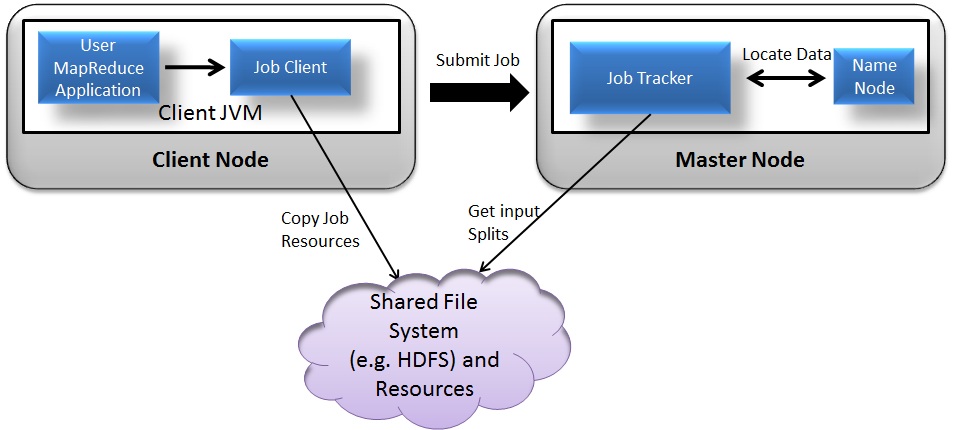
**MapReduce Architecture**

**MapReduce working:**

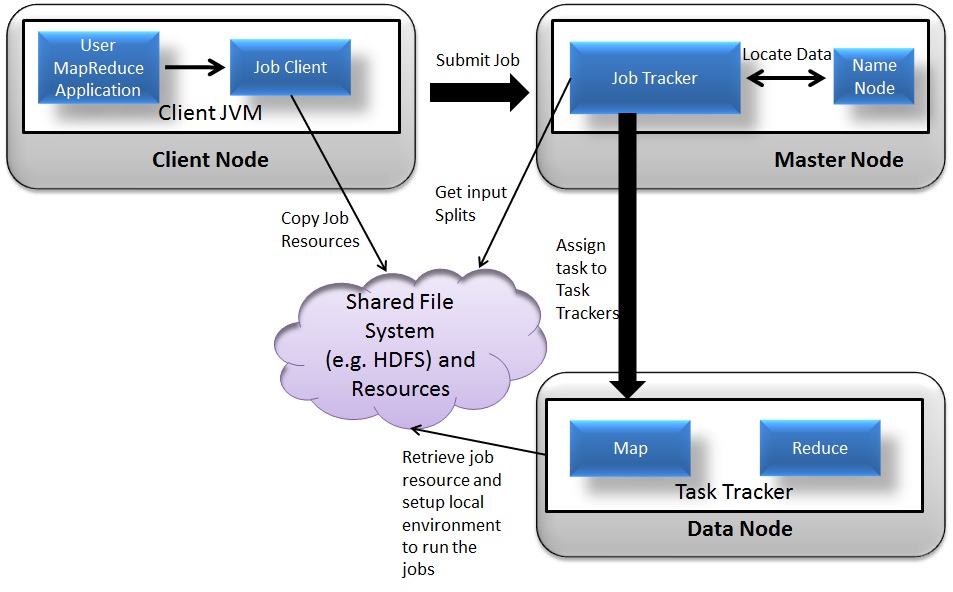
The entire process can be listed as follows:

1. Client applications submit jobs to the JobTracker.
2. The JobTracker talks to the NameNode to determine the location of the data
3. The JobTracker locates TaskTracker nodes with available slots at or near the data
4. The JobTracker submits the work to the chosen TaskTracker nodes.
5. The TaskTracker nodes are monitored. If they do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different TaskTracker.
6. A TaskTracker will notify the JobTracker when a task fails. The JobTracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may may even blacklist the TaskTracker as unreliable.
7. When the work is completed, the JobTracker updates its status.
8. Client applications can poll the JobTracker for information.
9. **Client submits MapReduce job to Job Tracker:**

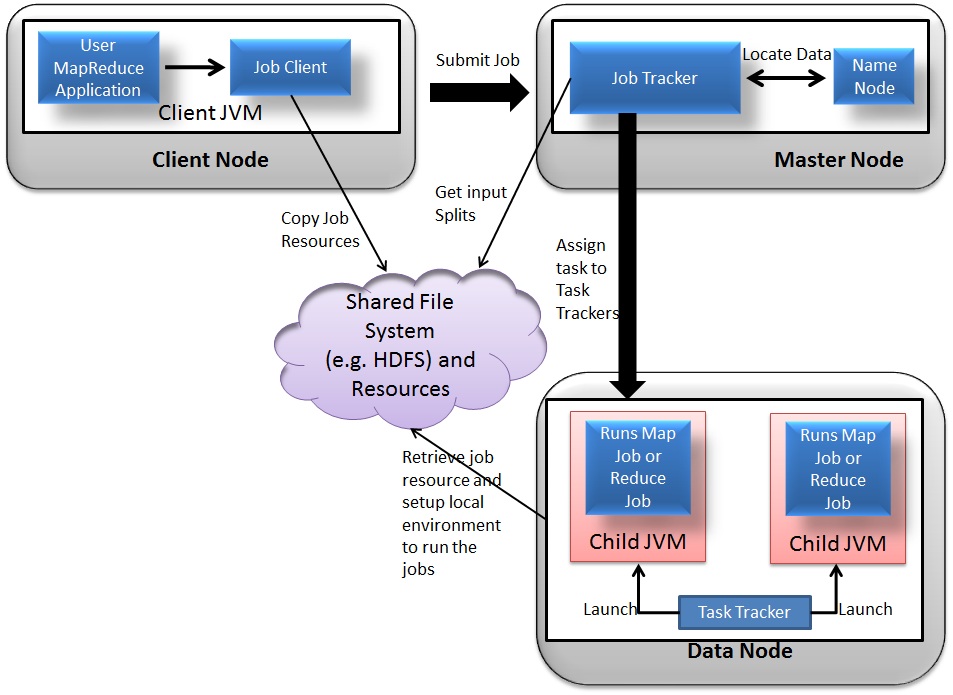
Whenever client/user submit map-reduce jobs, it goes straightaway to Job tracker. Client program contains all information like the map, combine and reduce function, input and output path of the data. **2. Job Tracker Manage and Control Job:**The JobTracker puts the job in a queue of pending jobs and then executes them on a FCFS(first come first serve) basis. The Job Tracker first determine the number of split from the input path and assign different map and reduce tasks to each TaskTracker in the cluster. There will be one map task for each split. The Job Tracker first determine the number of split from the input path and assign different map and reduce tasks to each TaskTracker in the cluster. There will be one map task for each split.



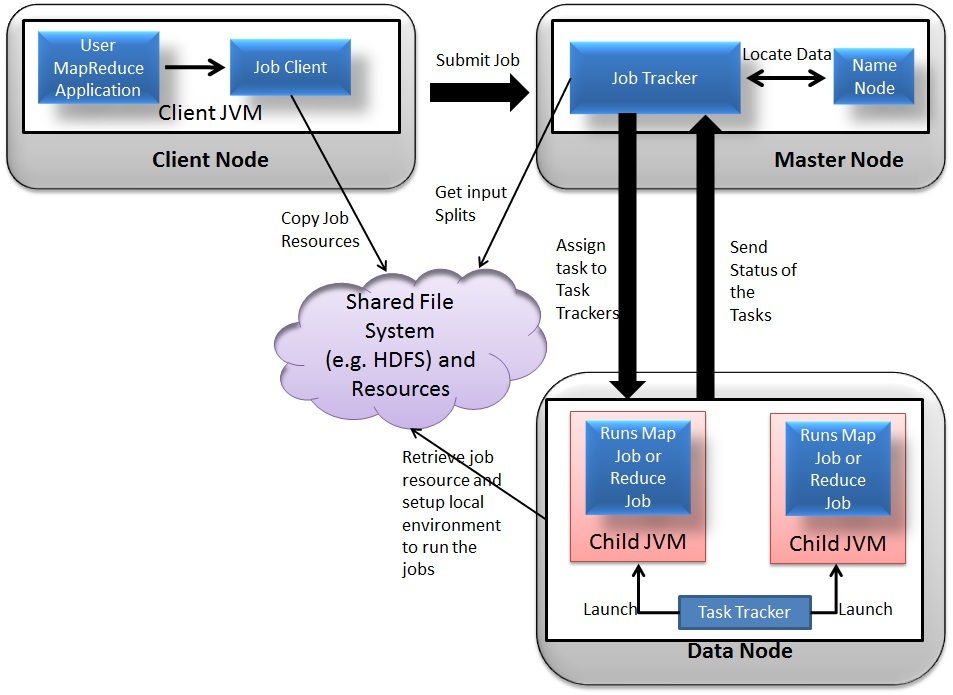
1. **Task Assignment to Task Tracker by Job Tracker:**The task tracker is pre-configured with a number of slots which indicates that how many task(in number) Task Tracker can accept. For example, a TaskTracker may be able to run two map tasks and two reduce tasks simultaneously. When the job tracker tries to schedule a task, it looks for an empty slot in the TaskTracker running on the same server which hosts the datanode where the data for that task resides. If not found, it looks for the machine in the same rack. There is no consideration of system load during this allocation.



1. **Task Execution by Task Tracker:**Now when the Task is assigned to Task Tracker, Task tracker creates local environment to run the Task. Now when the Task is assigned to Task Tracker, Task tracker creates local environment to run the Task. Task Tracker can also spawn multiple JVMs to handle many map or reduce tasks in parallel. TaskTracker actually initiates the Map or Reduce tasks and reports progress back to the JobTracker.

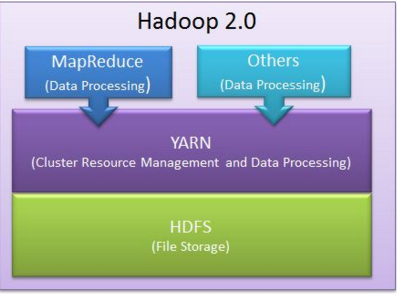


1. **Send notification to Job Tracker:**When all the map tasks are done by different task tracker they will notify the Job Tracker. Job Tracker then ask the selected Task Trackers to do the Reduce Phase

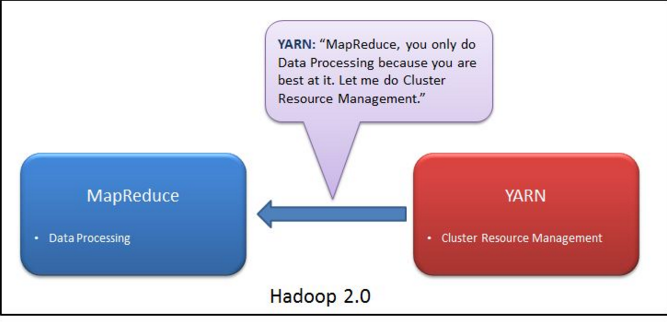


1. **Task recovery in failover situation:**Although there is single TaskTracker on each node, Task Tracker spawns off a separate Java Virtual Machine process to prevent the TaskTracker itself from failing if the running job(process) crashes the JVM due to some bugs defined in user written map reduce function.
2. **Monitor Task Tracker :**The TaskTracker nodes are monitored. A heartbeat is sent from the TaskTracker to the JobTracker every few minutes to check its status. If Task Tracker do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different TaskTracker. If Task Tracker do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different TaskTracker.
3. **Job Completion:**When the work is completed, the JobTracker updates its status. Client applications can poll the JobTracker for information.
4. YARN

YARN (**Y**et **A**nother **R**esource **N**egotiator) is a new component added in Hadoop 2.0

****

Cluster resource management means managing the resources of the Hadoop Clusters. And by resources we mean Memory, CPU etc.   
  
YARN took over this task of cluster management from MapReduce and MapReduce is streamlined to perform Data Processing only in which it is best.

****

YARN took over the task of cluster management from MapReduce and MapReduce is streamlined to perform Data Processing only in which it is best.

YARN has central resource manager component which manages resources and allocates the resources to the application. Multiple applications can run on Hadoop via YARN and all application could share common resource management.

**Advantage of YARN:**

1. **Yarn does efficient utilization of the resource:** There are no more fixed map-reduce slots. YARN provides central resource manager. With YARN, you can now run multiple applications in Hadoop, all sharing a common resource.
2. **Yarn can even run application that do not follow MapReduce model:** YARN decouples MapReduce's resource management and scheduling capabilities from the data processing component, enabling Hadoop to support more varied processing approaches and a broader array of applications. For example, Hadoop clusters can now run interactive querying and streaming data applications simultaneously with MapReduce batch jobs. This also streamlines MapReduce to do what is does best - process data.
3. **YARN is backward compatible:** This means that existing MapReduce job can run on Hadoop 2.0 without any change.
4. **No more JobTracker and TaskTracker needed in Hadoop 2.0:** JobTracker and TaskTracker has totally disappeared. YARN splits the two major functionalities of the JobTracker i.e. resource management and job scheduling/monitoring into 2 separate daemons (components).
5. Resource Manager
6. Node Manager(node specific)

Central Resource Manager and node specific Node Manager together constitutes YARN.

