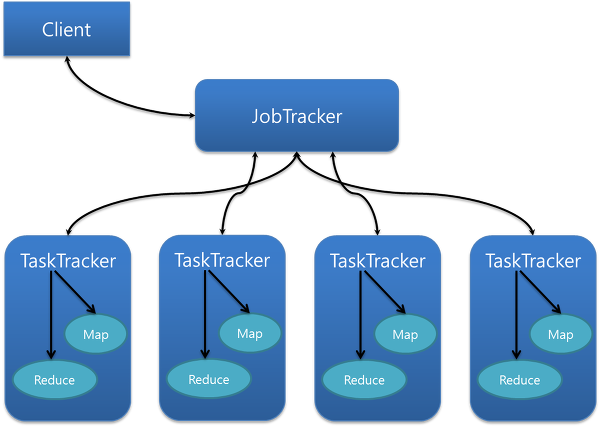
**Assignment : Hadoop Map Reduce**

1. **Map-reduce daemons (job tracker and task tracker) with diagrams.**



**Job Tracker –**

1. JobTracker process runs on a separate node and not usually on a DataNode.
2. JobTracker is an essential Daemon for MapReduce execution in MRv1. It is replaced by ResourceManager/ApplicationMaster in MRv2.
3. JobTracker receives the requests for MapReduce execution from the client.
4. JobTracker talks to the NameNode to determine the location of the data.
5. JobTracker finds the best TaskTracker nodes to execute tasks based on the data locality (proximity of the data) and the available slots to execute a task on a given node.
6. JobTracker monitors the individual TaskTrackers and the submits back the overall status of the job back to the client.
7. JobTracker process is critical to the Hadoop cluster in terms of MapReduce execution.
8. When the JobTracker is down, HDFS will still be functional but the MapReduce execution can not be started and the existing MapReduce jobs will be halted.

**TaskTracker –**

1. TaskTracker runs on DataNode. Mostly on all DataNodes.
2. TaskTracker is replaced by Node Manager in MRv2.
3. Mapper and Reducer tasks are executed on DataNodes administered by TaskTrackers.
4. TaskTrackers will be assigned Mapper and Reducer tasks to execute by JobTracker.
5. TaskTracker will be in constant communication with the JobTracker signalling the progress of the task in execution.
6. TaskTracker failure is not considered fatal. When a TaskTracker becomes unresponsive, JobTracker will assign the task executed by the TaskTracker to another node.

**4) YARN architecture and YARN Daemons**

In Hadoop 1.x Architecture JobTracker daemon was carrying the responsibility of Job scheduling and Monitoring as well as was managing resource across the cluster. And TaskTracker daemon was executing map reduce tasks on the slave nodes. YARN has divided the responsibilities of JobTracker to two processes ResourceManager and ApplicationMaster and instead ofTaskTracker is using NodeManager daemon for map reduce task execution.

**1) ResourceManager**

* This daemon process resides on the Master Node (not necessarily on NameNode of Hadoop)
* Responsible for,
  + Managing resources scheduling for different compute applications in an optimum way
  + Coordinating with two process on master node, **Scheduler** and **ApplicationManager**

**Scheduler**

* This daemon process resides on the Master Node (runs along with ResourceManager daemon )
* Responsible for,
  + Scheduling the job execution as per submission request received by ResourceManager
  + Allocating resources to applications submitted to the cluster
  + Coordinating with ApplicationManager daemon and keeping track of resources of running applications

**ApplicationManager**

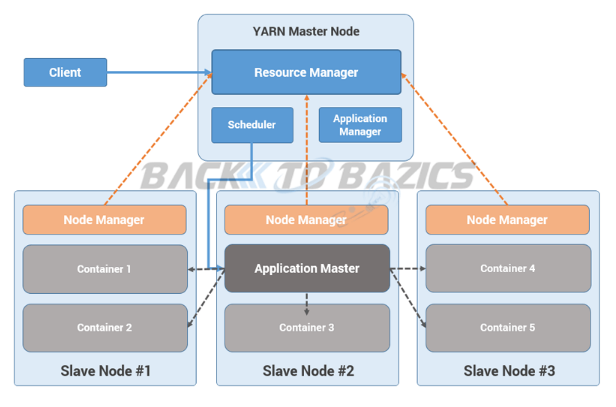
* This daemon process resides on the Master Node (runs along with ResourceManager daemon )
* Responsible for,
  + Helping Scheduler daemon to keeps track of running application by coordination
  + Accepting job submissions from client
  + Negotiating first container for executing application specific task with suitable ApplicationMaster on slave node

**2) NodeManager**

* This daemon process resides on the slave nodes (runs along with DataNode daemon)
* Responsible for,
  + Managing and executing containers
  + Monitoring resource usage (i.e. usage of memory, cpu, network etc..) and reporting it back to ResourceManager daemon
  + Periodically sending heart-bits to ResourceManager for its health status update

**3) ApplicationMaster**

* This daemon process runs on the slave node (along with the NodeManager daemon)
* It is per application specific library works with NodeManager to execute the task
* The instance of this daemon is per application, which means in case of multiple jobs submitted on cluster, it may have more than one instances of ApplicationMaster on slave nodes
* Responsible for,
  + Negotiating suitable resource containers on slave node from ResourceManager
  + Working with one or multiple NodeManagers to monitor task execution on slave nodes



**5) Explanation and working of YARN**

**Step 1:**  Job/Application(which can be MapReduce, Java/Scala Application, DAG jobs like Apache Spark etc..) is submitted by the YARN client application to the ResourceManager daemon along with the command to start the ApplicationMaster on any container atNodeManager

**Step 2:**  ApplicationManager process on Master Node validates the job submission request and hand it over to Scheduler process for resource allocation

**Step 3:**  Scheduler process assigns a container for ApplicationMaster on one slave node

**Step 4:**  NodeManager daemon starts the ApplicationMaster service within one of its container using the command mentioned in Step 1, hence ApplicationMaster is considered to be the first container of any application

**Step 5:**  ApplicationMaster negotiates the other containers from ResourceManager by providing the details like location of data on slave nodes, required cpu, memory, cores etc..

**Step 6:**  ReourceManager allocates the best suitable resources on slave nodes and responds to ApplicationMaster with node details and other details

**Step 7:**  Then, ApplicationMaster send requests to NodeManagers on suggested slave nodes to start the containers

**Step 8:**  ApplicationMaster than manages the resources of requested containers while job execution and notifies theResourceManager when execution is completed

**Step 9:**  NodeManagers periodically notify the ResourceManager with the current status of available resources on the node which information can be used by scheduler to schedule new application on the clusters

**Step 10:**  In case of any failure of slave node ResourceManager will try to allocate new container on other best suitable node so thatApplicationMaster can complete the process using new container

**6) YARN Vs. Job tracker and task tracker**

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.

**8) Diagram of hadoop ecosystem.**

