MANAGING RESOURCES AND APPLICATIONS WITH HADOOP - YARN

(YET ANOTHER RESOURCE NEGOTIATOR)

Limitations of Hadoop 1.0 Architecture

- Single Namenode is responsible for managing entire namespace for Hadoop Cluster.
- It has a restricted processing model which is suitable for batch-oriented MapReduce jobs.
- Hadoop MapReduce is not suitable for interactive analysis.
- Hadoop 1.0 is not suitable for machine learning algorithms, graphs, and other memory intensive algorithms.
- **MapReduce** is responsible for **cluster resource management and data processing**.
- Resource utilization problems also there.
- Name Node saves all its file metadata in main memory (**HDFS federation**)

Hadoop2:YARN

- HDFS consists of 2 major components:
  1. Namespace : File operations
  2.Block Storage Service: Handles cluster management and replication

HDFS features:

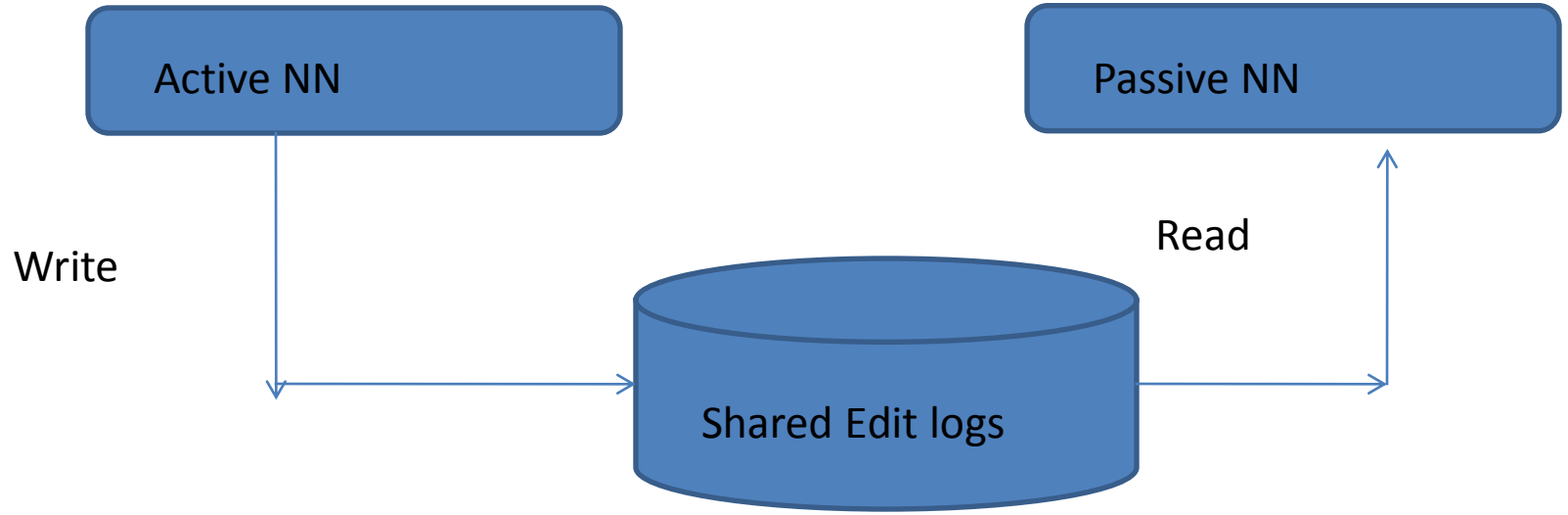
1. Horizontally Scalability
2.High availability

HDFS Federation:

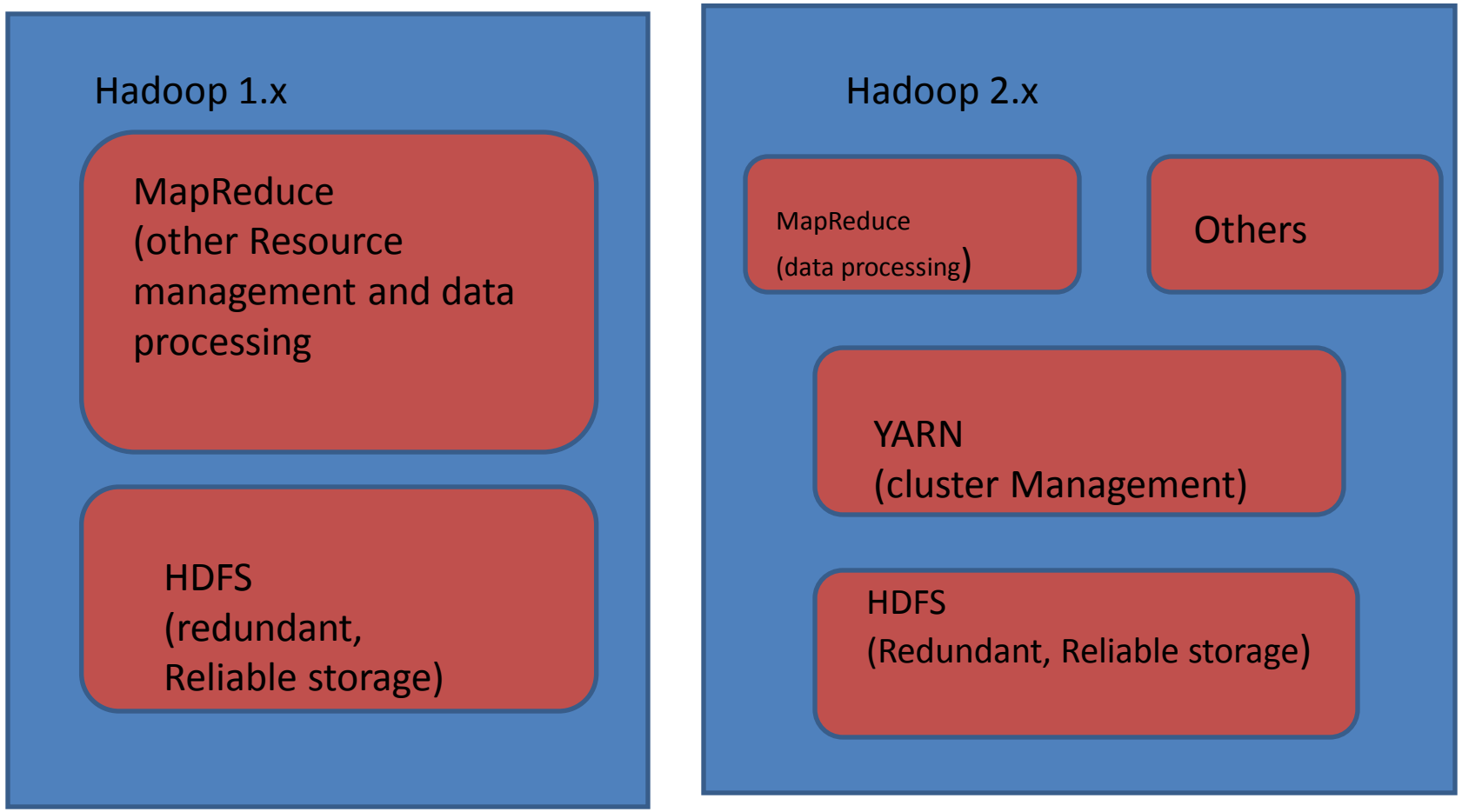
1. It uses multiple independent name nodes for horizontal scalability.
2. Data nodes are common storage for blocks and shared by all NN.
3. All DN nodes must registers with each NN in clusters.
4. High Availability is achieved by Passive-standby NN
5. Active-Passive NN handles failover automatically

Active-Passive name node interactions

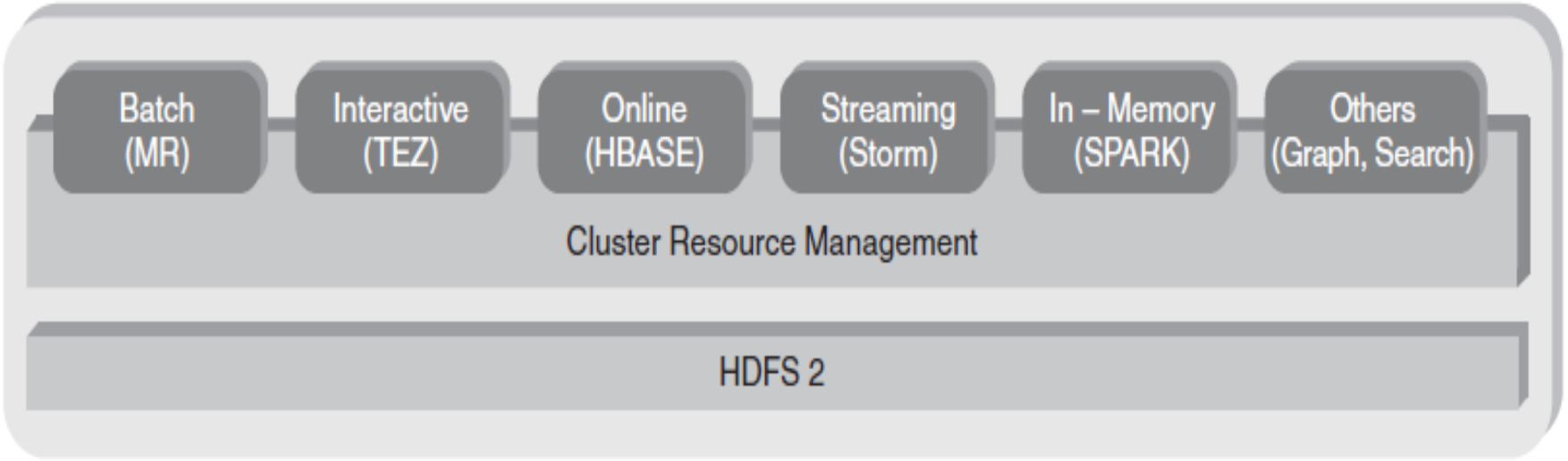
- All namespace edits are recorded to shared NFS and there is a single writer at any time.
- Passive NN read edits from shared storage and keeps updated information.
- In case Active NN fails then Passive NN become Active NN automatically.



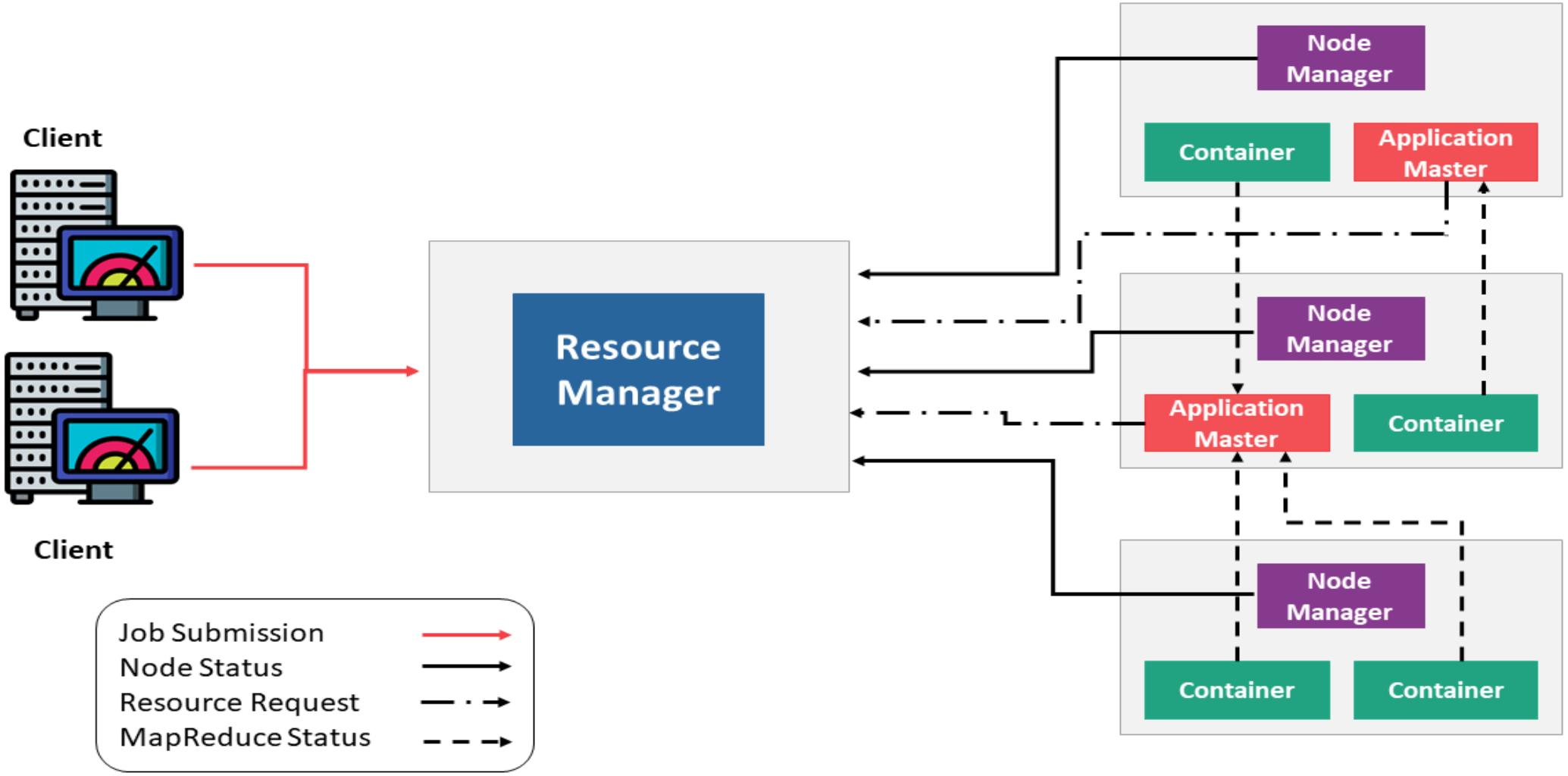
Hadoop 1.x Vs. Hadoop2.x



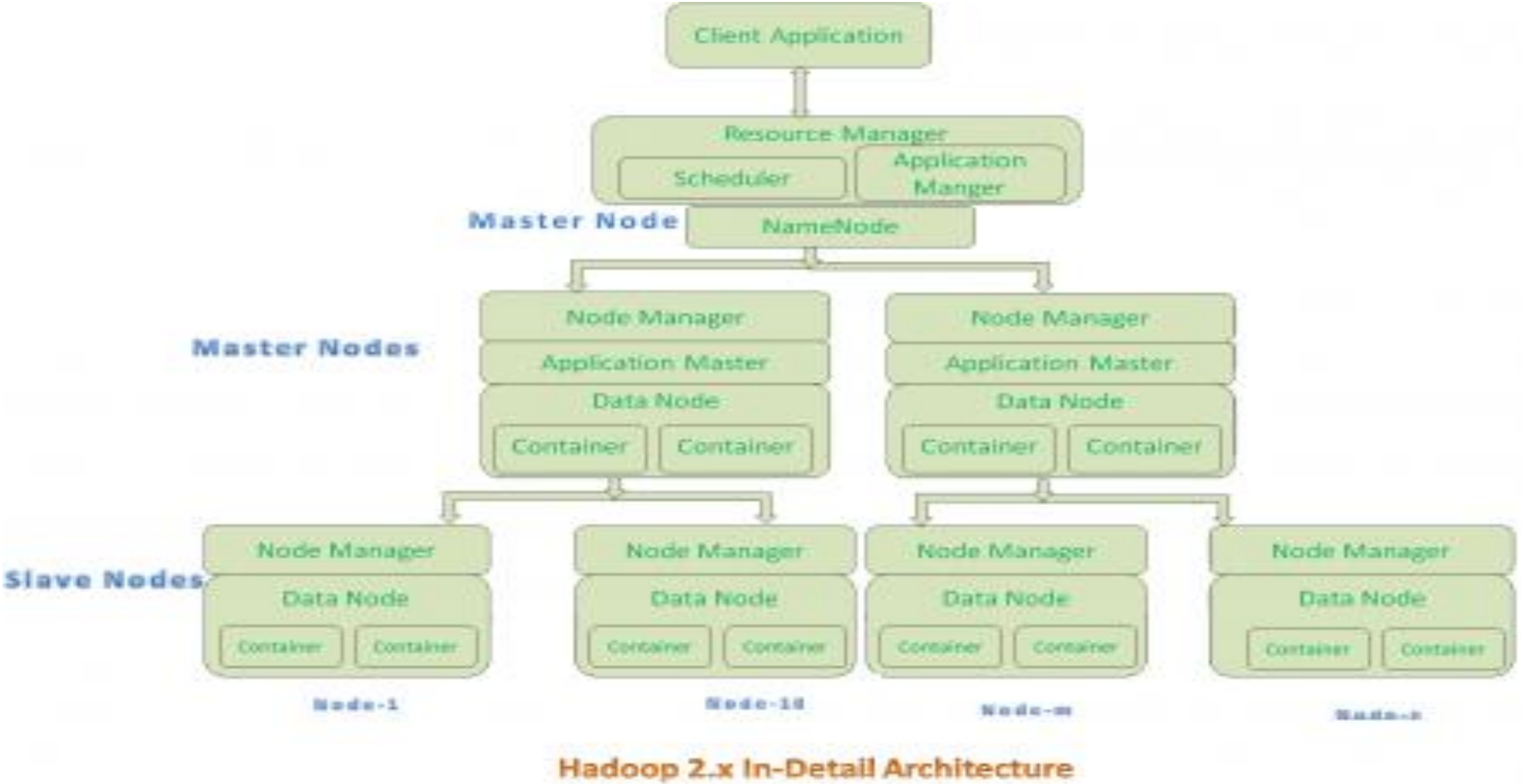
Hadoop 2 YARN: Taking Hadoop beyond Batch



Hadoop 2.x Architecture



Yarn Detail architecture



YARN Components

- **Resource Manager**
- It is the ultimate authority in resource allocation.
- On receiving the processing requests, it passes parts of requests to corresponding node managers accordingly, where the actual processing takes place.
- It is the arbitrator of the cluster resources and decides the allocation of the available resources for competing applications.
- Optimizes the cluster utilization like keeping all resources in use all the time against various constraints such as capacity guarantees, fairness, and SLAs.
- It has two major components:

    a) Scheduler

    b) Application Manager

YARN Components

- *a) Scheduler*
- The scheduler is responsible for allocating resources to the various running applications subject to constraints of capacities, queues etc.
- It is called a pure scheduler in ResourceManager, which means that it does not perform any monitoring or tracking of status for the applications.
- If there is an application failure or hardware failure, the Scheduler does not guarantee to restart the failed tasks.
- Performs scheduling based on the resource requirements of the applications.
- It has a pluggable policy plug-in, which is responsible for partitioning the cluster resources among the various applications. There are two such plug-ins: **Capacity Scheduler** and **Fair Scheduler**, which are currently used as Schedulers in ResourceManager.

YARN Components

- *b) Application Manager*
- It is responsible for accepting job submissions.
- Negotiates the first container from the Resource Manager for executing the application specific Application Master.
- Manages running the Application Masters in a cluster and provides service for restarting the Application Master container on failure.

YARN Components

- **Node Manager**
- It takes care of individual nodes in a Hadoop cluster and manages user jobs and workflow on the given node.
- It registers with the Resource Manager and sends heartbeats with the health status of the node.
- Its primary goal is to manage application containers assigned to it by the resource manager.
- It keeps up-to-date with the Resource Manager.
- Application Master requests the assigned container from the Node Manager by sending it a Container Launch Context(CLC) which includes everything the application needs in order to run. The Node Manager creates the requested container process and starts it.
- Monitors resource usage (memory, CPU) of individual containers.
- Performs Log management.
- It also kills the container as directed by the Resource Manager.

YARN Components

- **Application Master**
- An application is a single job submitted to the framework. Each such application has a unique Application Master associated with it which is a framework specific entity.
- It is the process that coordinates an application's execution in the cluster and also manages faults.
- Its task is to negotiate resources from the Resource Manager and work with the Node Manager to execute and monitor the component tasks.
- It is responsible for negotiating appropriate resource containers from the ResourceManager, tracking their status and monitoring progress.
- Once started, it periodically sends heartbeats to the Resource Manager to affirm its health and to update the record of its resource demands.

YARN Components

- **Container**
- It is a collection of physical resources such as RAM, CPU cores, and disks on a single node.
- YARN containers are managed by a container launch context which is container life-cycle(CLC). This record contains a map of environment variables, dependencies stored in a remotely accessible storage, security tokens, payload for Node Manager services and the command necessary to create the process.
- It grants rights to an application to use a specific amount of resources (memory, CPU etc.) on a specific host.

Application Submission in YARN

1) Submit the job

2) Get Application ID

3) Application Submission Context

4 a) Start Container Launch

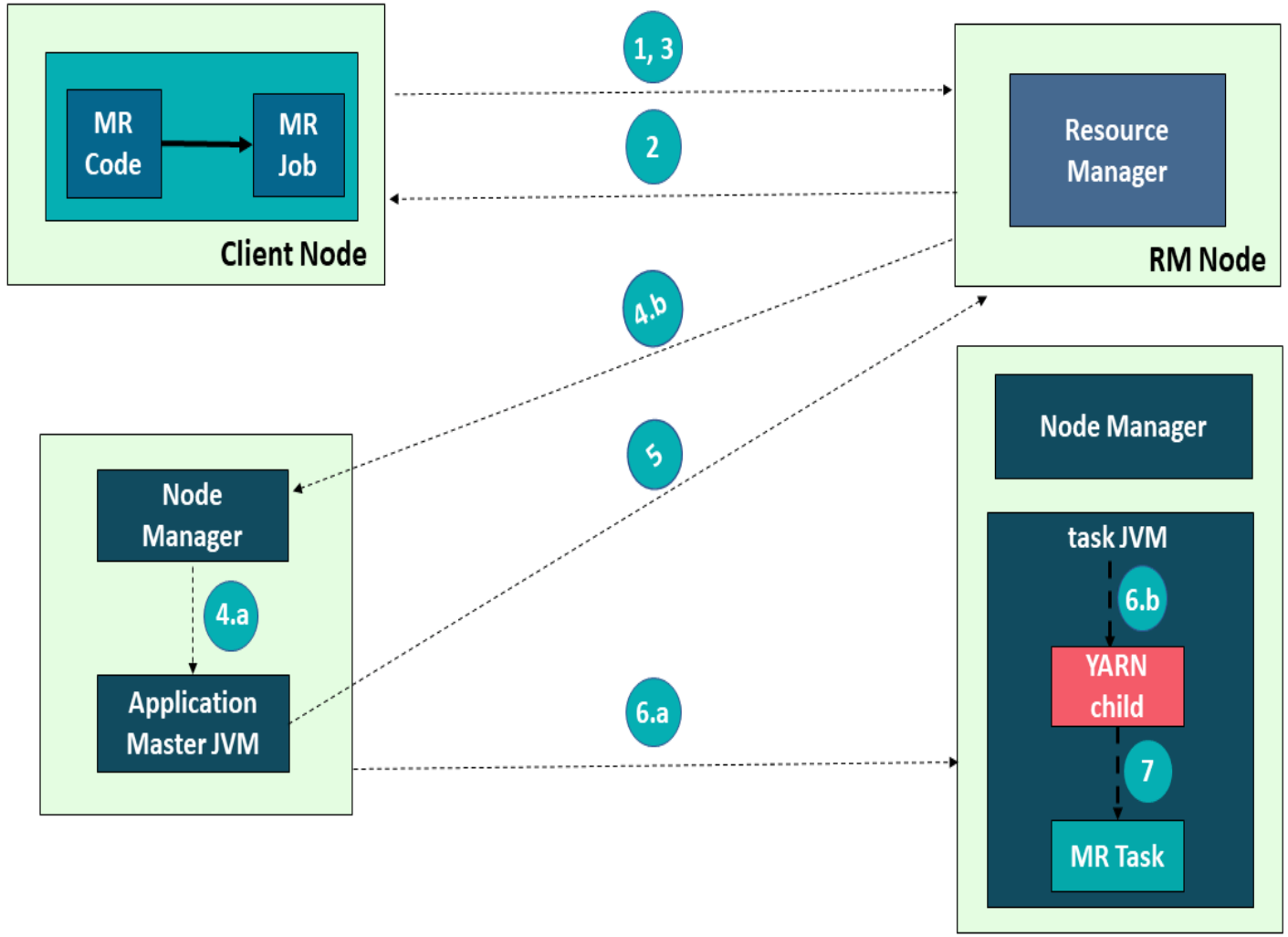
b) Launch Application Master

5) Allocate Resources

6 a) Container

b) Launch

7) Execute



Application Workflow in Hadoop YARN

1. Client submits an application which includes necessary specification to launch application specific – APPLICATION MASTER itself.
2. Resource Manager allocates a container to start Application Master
3. Application Master registers with Resource Manager
4. Application Master asks containers from Resource Manager
5. On successful allocation Application Master notifies Node Manager to launch containers
6. Node manager executes Application code in the container
7. Client contacts Resource Manager/Application Manager to monitor application’s status
8. Application Manager unregisters with Resource Manager