Overview

Big Data in Apache™ Hadoop®

- HDFS
- MapReduce in Hadoop
- YARN

https://hadoop.apache.org

Apache Hadoop - Historical Background

- 2003: Google publishes its cluster architecture & DFS (GFS)
- 2004: MapReduce as a new programming model is introduced by Google, working on top of GFS
  - written in C++ as a closed-source project
- 2006: Apache & Yahoo! publish Hadoop & HDFS
  - Java implementation of Google MapReduce and GFS
- 2008: Hadoop becomes an independent Apache project

- **Today**: Hadoop widely used as a general-purpose storage and analysis platform for big data

Big Data in Hadoop

Apache Hadoop - Google: The Data Challenge

Keynote Speech, Jeffrey Dean (Google), 2006, PACT'06:
Problems:
- 20+ billion web pages x 20KB = 400+ terabytes
- One computer can read 30-35 MB/sec from disk
  - —> four months to read the web
- takes even longer to 'do' something with the data
- But: same problem with 1000 machines, < 3 hours

MapReduce CACM'08 article:
- 100,000 MapReduce jobs executed in Google every day
- Total data processed > 20 PB per day

Apache Hadoop

- open-source SW for reliable, scalable, distributed computing
- Some provided modules:
  - **Hadoop Common**:
    - common utilities for distributed filesystems and general I/O (serialization, Java RPC, persistent data structures)
  - **Hadoop Distributed File System** (HDFS):
    - A distributed file system that provides high-throughput access to application data
  - **Hadoop YARN**:
    - framework for job scheduling and cluster resource management
  - **Hadoop MapReduce**:
    - distributed data-processing model and execution environment running on large clusters of commodity machines

Apache Hadoop

- Tools within Hadoop

Tool	*Description*
HBase	***Distributed, column-oriented database***
Hive	***Distributed data warehouse***
Pig	***Higher-level data flow language and parallel execution framework***
ZooKeeper	***Distributed coordination service***
Avro	***Data serialisation system***
Sqoop	***Tool for bulk data transfer between structured data stores and HDFS***
Oozie	***Complex job workflow service***
Chukwa	***System for collecting management data***
Mahout	***Machine learning and data mining library***
BigTop	***Packaging and testing***
Avro	***serialization system for cross-language RPC and persistent data storage***

Apache Hadoop - Common Use Cases

- Log Data Analysis
  - most common: write once & read often
- Fraud Detection
- Risk Modeling
- Social Sentiment Analysis
- Image Classification
- Graph Analysis
- ...

Overview

Big Data in Apache™ Hadoop®

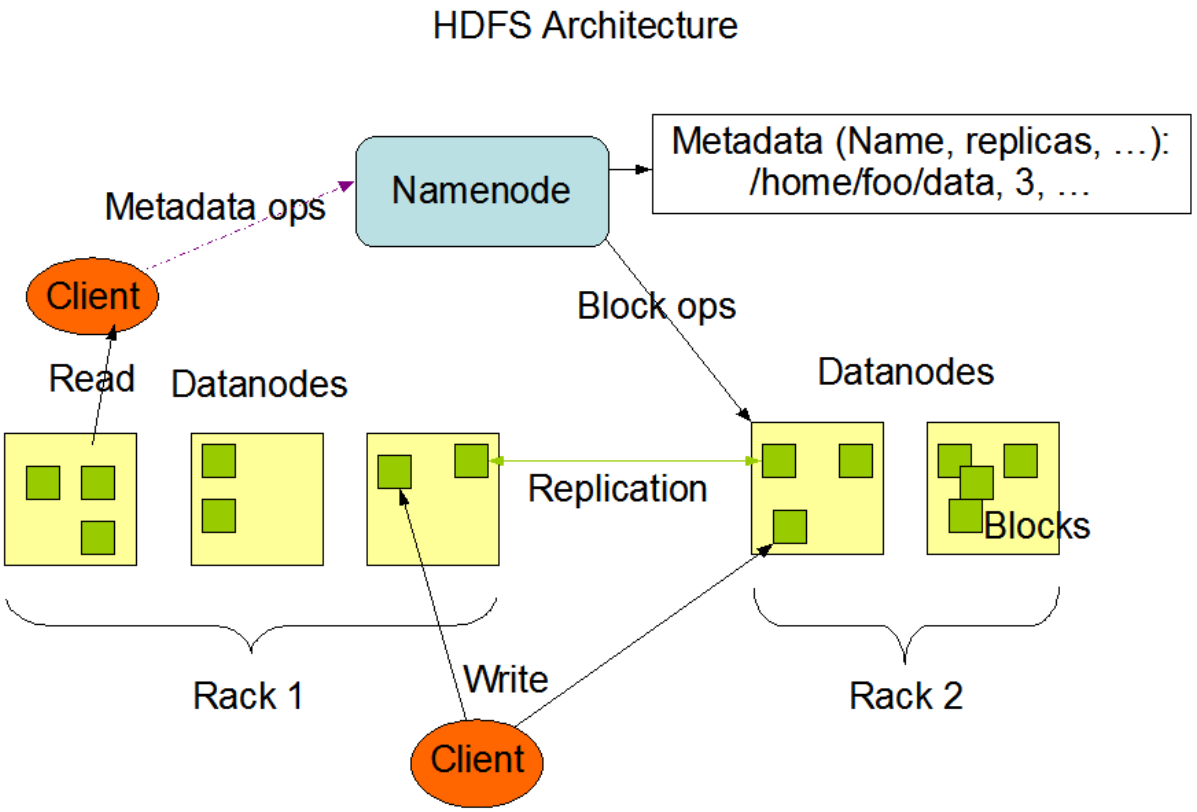
- HDFS
- MapReduce in Hadoop
- YARN

https://hadoop.apache.org

Apache Hadoop - Architecture of HDFS

- HDFS: A distributed file system that provides high-throughput access to application data
- HDFS has a master/slave architecture
  - **NameNode** as master
    - arbitrator and repository for all HDFS metadata
    - manages the file system namespace
    - mapping of blocks to DataNodes
    - regulates access to files by clients
  - **DataNodes** as clients:
    - manage storage attached to the nodes that they run on
    - storing “blocks” of files
    - responsible for serving read and write requests from the clients
    - perform block creation,deletion and replication upon instruction from the NameNode

Apache Hadoop - Hadoop Distributed File System (HDFS)



Source: http://hortonworks.com/hadoop/hdfs/#section_2

Apache Hadoop - Data Storage Operations on HDFS

- **Characteristics:**
  - Write One, Read Often Model
  - Content of individual files cannot be modified, but we can append new data at the end of a file

- **Operations:**
  - create a new file
  - delete a file
  - rename a file
  - append content to the end of a file
  - modify file attributes (owner, read, write)

Apache Hadoop - Partitioning the input data

HDFS Blocks

- File is divided into blocks (default:64 MB) and duplicated in multiple nodes (default: 3 replicas)
  - —> Fault tolerance
- Dividing files into blocks is common for a FS, e.g. default block size in Linux is 4KB.
  - —> Difference of HDFS is the **scale**
- Hadoop was designed to operate at the petabyte scale
- Every data block stored in HDFS has its own metadata and needs to be tracked by a central server
- files in HDFS are write-once and have strictly one writer at any time

Apache Hadoop - Partitioning the input data

HDFS Blocks

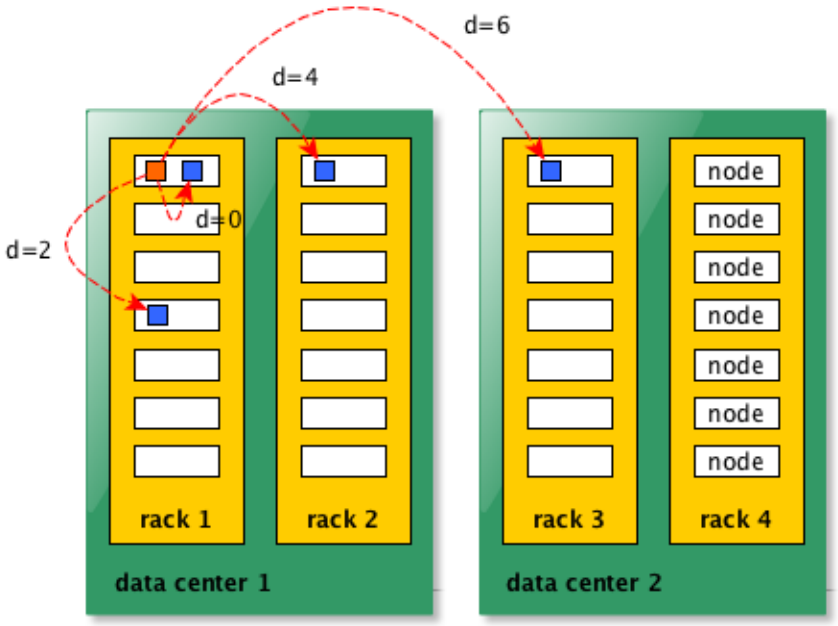
- Network Topology and Distance

process on the same node
- distance(/d1/r1/n1, /d1/r1/n1) = 0

different nodes / same rack
- distance(/d1/r1/n1, /d1/r1/n2) = 2

different racks/same datacenter
- distance(/d1/r1/n1, /d1/r2/n3) = 4

different datacenters
- distance(/d1/r1/n1, /d2/r3/n4) = 6

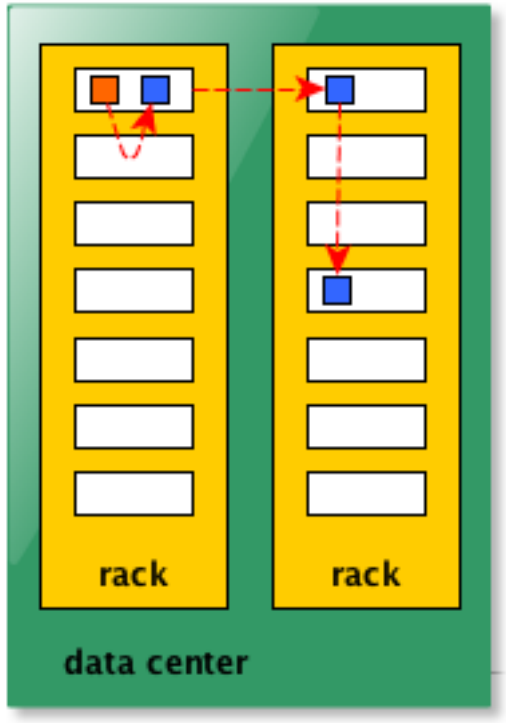


Apache Hadoop - Partitioning the input data

HDFS Blocks

Hadoop's default **replica placement** strategy:

- **1st** replica:
  - same node as the client
  - if node outside cluster, choose a random node
- **2nd** replica:
  - *off-rack*, chosen at random
- **3rd** replica:
  - same rack as 2nd, but on a different node, chosen at random
- further replicas:
  - random nodes; avoid placing too many replica on same rack



Apache Hadoop - HDFS Robustness

- **Data Disk Failure, Heartbeats** and **Re-replication**
- **Cluster Rebalancing**
  - move data from one DataNode to another if free space falls below a certain threshold
- **Data Integrity**
  - checksum checking of each block of a file
- **Metadata Disk Failure**
  - replica of nameNode + copies of log files
- **Snapshots**
  - support of storing a copy of data at a particular instant of time

Data Disk Failure, Heartbeats and Re-replication

- Each DataNode sends a Heartbeat message to the NameNode periodically.
- (subsets of) DataNodes may lose connectivity with the NameNode
  → detected by absence of Heartbeats
- Affected DataNodes marked as dead → no new IO requests from NameNode.
- Any data that was registered to a dead DataNode is not available to HDFS any more.
- DataNode death may cause the replication factor of some blocks to fall below their specified value
  → re-replication (initiated by NameNode) needed.
- re-replication also required if the replication factor of a file has been increased.

Cluster Rebalancing

- The HDFS architecture is compatible with data rebalancing schemes.
- A scheme might automatically move data from one DataNode to another if the free space on a DataNode falls below a certain threshold.
- In the event of a sudden high demand for a particular file, a scheme might dynamically create additional replicas and rebalance other data in the cluster.
- These types of data rebalancing schemes are not yet implemented.

Big Data in Hadoop

- **Data Integrity**
  - It is possible that a block of data fetched from a DataNode arrives corrupted (e.g. because of faults in a storage device, network faults, or buggy software.
  - The HDFS client software implements **checksum checking** on the contents of HDFS files. When a client creates an HDFS file, it computes a checksum of each block of the file and stores these checksums in a separate hidden file in the same HDFS namespace.
  - When a client retrieves file contents it verifies that the data it received from each DataNode matches the checksum stored in the associated checksum file. If not, then the client can opt to retrieve that block from another DataNode that has a replica of that block.

- **Metadata Disk Failure**
  - FsImage and EditLog are central data structures of HDFS.
  - Corruption of these files can cause the HDFS instance to be non-functional.
    → maintaining multiple copies of the FsImage and EditLog.
  - Updates to each FsImage or EditLog instance only synchronously → degrades the rate of namespace transactions per second (but HDFS applications are not metadata intensive)
  - When a NameNode restarts, it selects the latest consistent FsImage and EditLog to use.
  - The NameNode machine is a single point of failure for an HDFS cluster. If the NameNode machine fails, manual intervention is necessary. Currently, automatic restart and failover of the NameNode software to another machine is not supported.

Big Data in Hadoop

- **Snapshots**
  - Snapshots support storing a copy of data at a particular instant of time.
  - One usage of the snapshot feature may be to roll back a corrupted HDFS instance to a previously known good point in time.
  - HDFS does not currently support snapshots but will in a future release.

Overview

Big Data in Apache™ Hadoop®

- HDFS
- MapReduce in Hadoop
- YARN

https://hadoop.apache.org

Apache Hadoop - HDFS/MapReduce layer composition

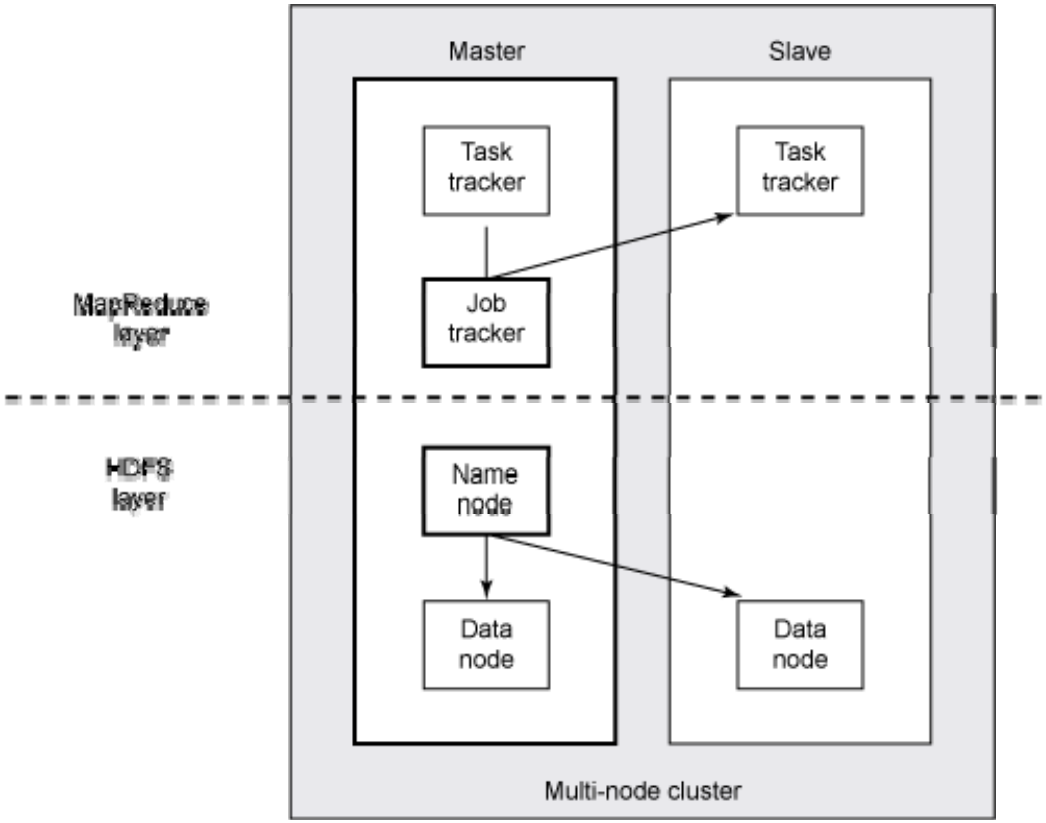
Bringing it all together…

MapReduce tasks are divided into *trackers*:

- Master: JobTracker
- Slave: TaskTracker

HDFS Tasks divided into *nodes*:

- Master: NameNode
- Slaves: DataNodes



Apache Hadoop - HDFS/MapReduce layer composition

Bringing it all together…

JobTracker:

- coordinates all the jobs running on a system
- scheduling tasks to run on TaskTrackers
- keeps record of the overall progress of each job
- on Task failure: reschedule tasks on different TaskTracker

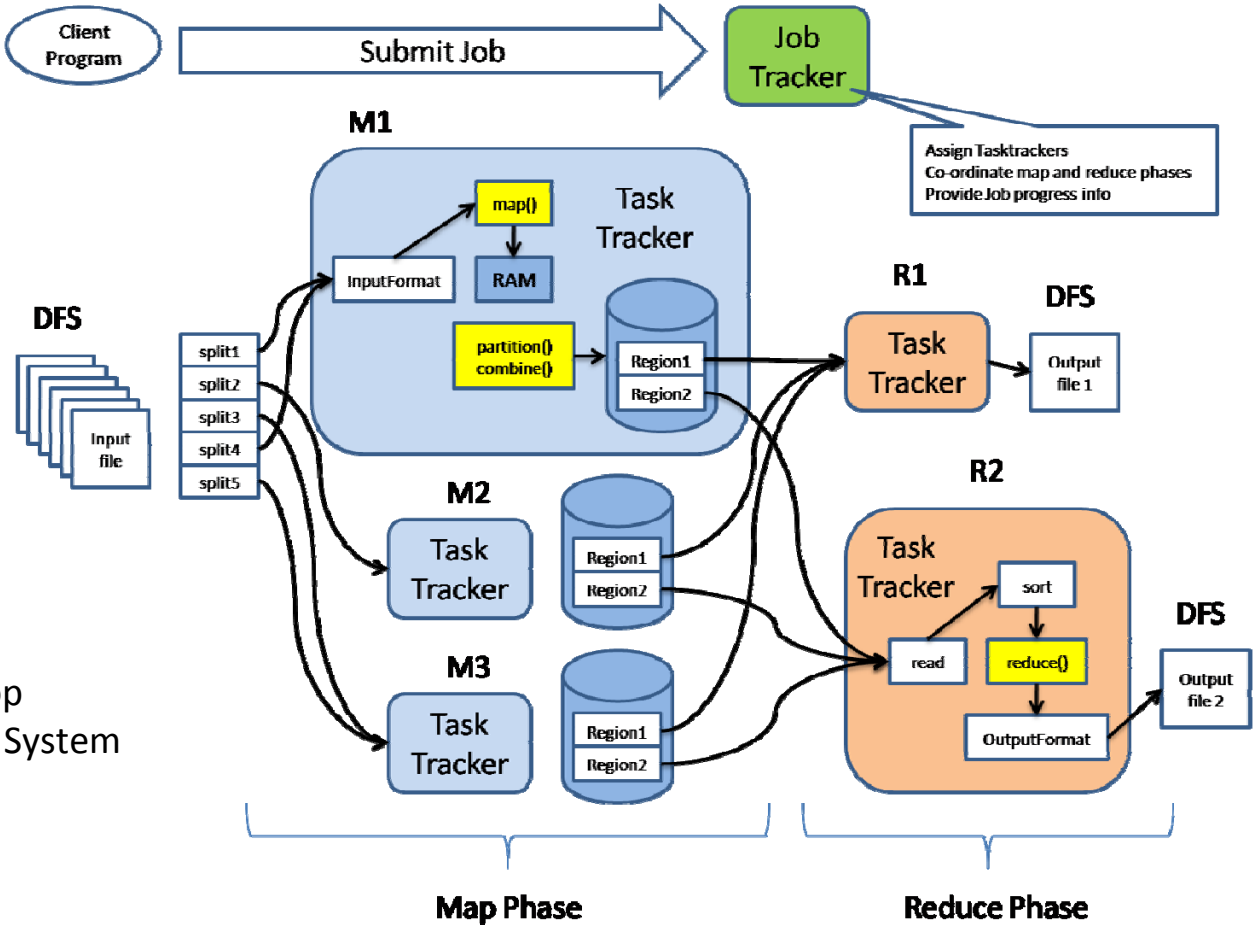
TaskTracker:

- execute tasks
- send progress reports to JobTracker

Apache Hadoop - Workflow of MapReduce in Hadoop

- **1.** Client submits an application request to the JobTracker
- **2.** JobTracker determines processing resources to execute the entire application, e.g. selection of TaskTrackers based on their proximity to the data source
- **3.** JobTracker identifies state of the slave nodes and queues all map tasks and reduce tasks for execution
- **4.** When processing slots become available on slave nodes, map tasks are deployed
- **5.** JobTracker monitors task progress. On failure, the task is restarted on next available slot.
- **6.** When map tasks have finished, reduce tasks process the interim results sets
- **7.** The result set is returned to the client application

Apache Hadoop - MapReduce in Hadoop



DFS: e.g. Hadoop Distributed File System (HDFS)

Source: https://www.ibm.com/developerworks/cloud/library/cl-openstack-deployhadoop/

Apache Hadoop - Limitations of MapReduce

- MapReduce is a successful batch-oriented programming model

- Increasing demand for additional processing modes:
  - Graph Analysis
  - Stream data processing
  - Text Analysis
  - —>Demand is growing for real-time and ad-hoc analysis
  - —> Analysts with specific queries including only subsets of data and need an instant response

- —> Solution of the Hadoop/MapReduce-World: **YARN**

Overview

Big Data in Apache™ Hadoop®

- HDFS
- MapReduce in Hadoop
- YARN

https://hadoop.apache.org

Apache Hadoop - MapReduce 2.0 or YARN

- **YARN: Yet Another Resource Negotiator**
- *idea*: split up the two major functionalities of the JobTracker (resource management and job scheduling/monitoring), into separate daemons:
  - ResourceManager
  - ApplicationMaster

- MasterNode runs the ResourceManager and on each slave runs a NodeManager —> Data-computation framework.

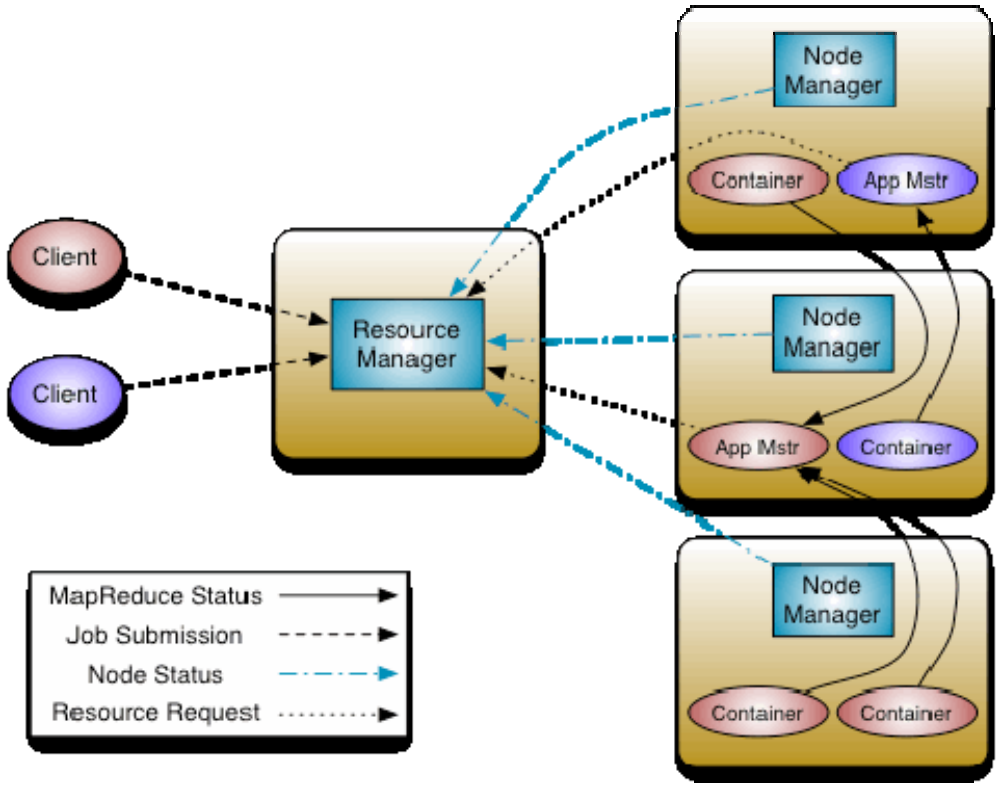
- ApplicationMaster works with the NodeManager(s) to execute and monitor the tasks

Apache Hadoop - Workflow of MapReduce 2.0 or YARN

General workflow of a YARN application:

- **1.** Client submits application to Resource Manager
- **2.** ResourceManager asks any NodeManager to create an ApplicationMaster, which register with the Resource Manager
- **3.** ApplicationMaster determines how many resources are needed and requested the necessary resources from the ResourceManager
- **4.** ResourceManager accepts the requests and queue up
- **5.** As the requests resources become available on slave nodes, the ResourceManager grants the ApplicationMaster requirements for containers on specific slave nodes

Apache Hadoop - MapReduce 2.0 or YARN



Source: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html