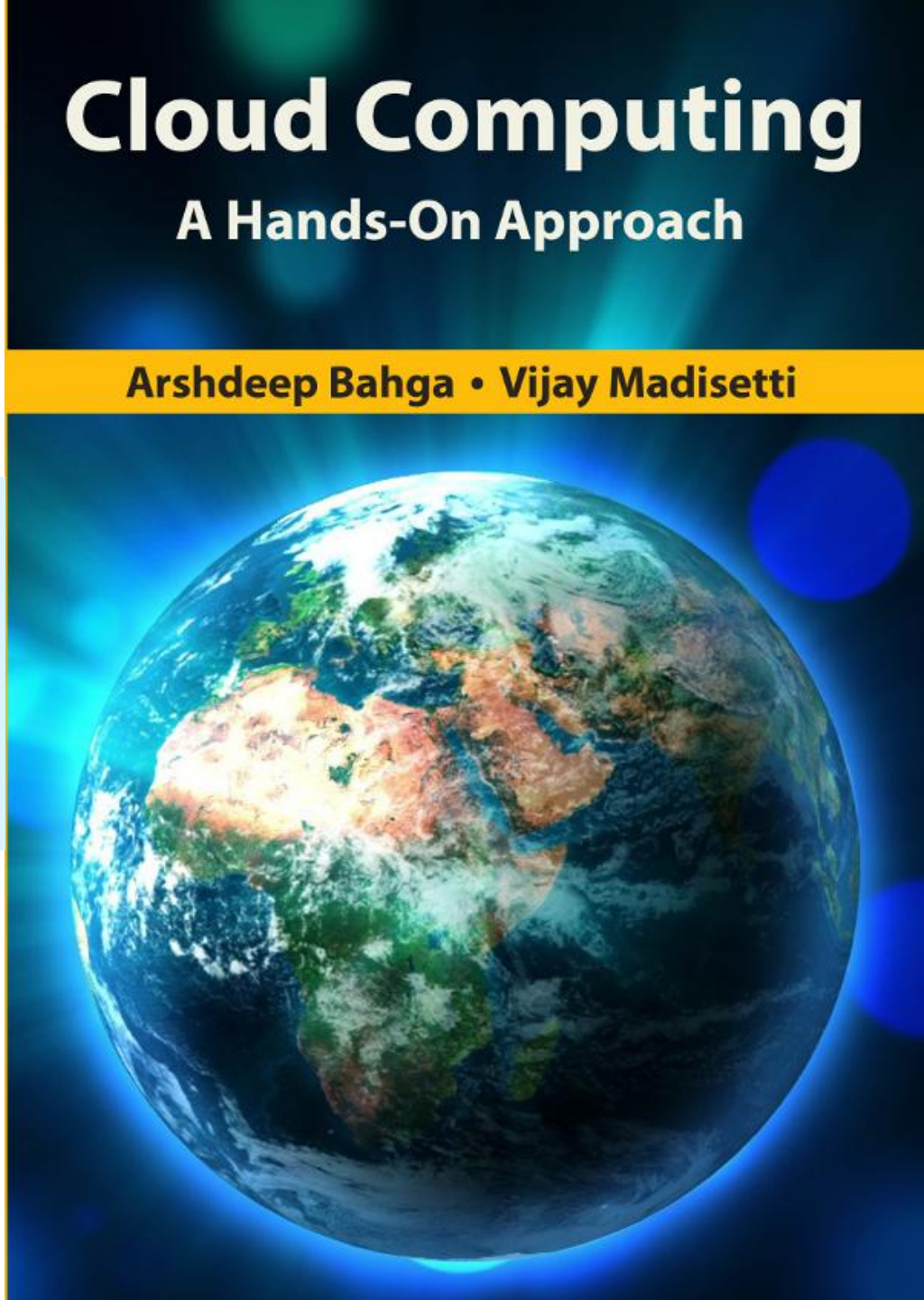
Chapter 4

Hadoop & MapReduce



Outline

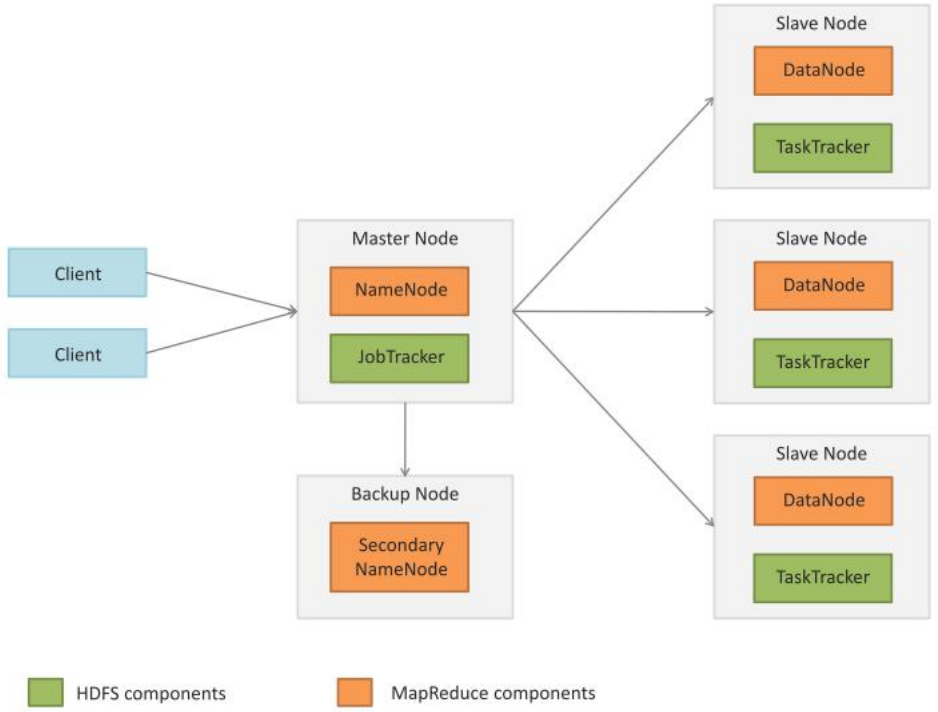
- Overview of Hadoop ecosystem
- MapReduce architecture
- MapReduce job execution flow
- MapReduce schedulers

Hadoop Ecosystem

- Apache Hadoop is an open source framework for distributed batch processing of big data.
- Hadoop Ecosystem includes:
  - Hadoop MapReduce
  - HDFS
  - YARN
  - HBase
  - Zookeeper
  - Pig
  - Hive
  - Mahout
  - Chukwa
  - Cassandra
  - Avro
  - Oozie
  - Flume
  - Sqoop

Apache Hadoop

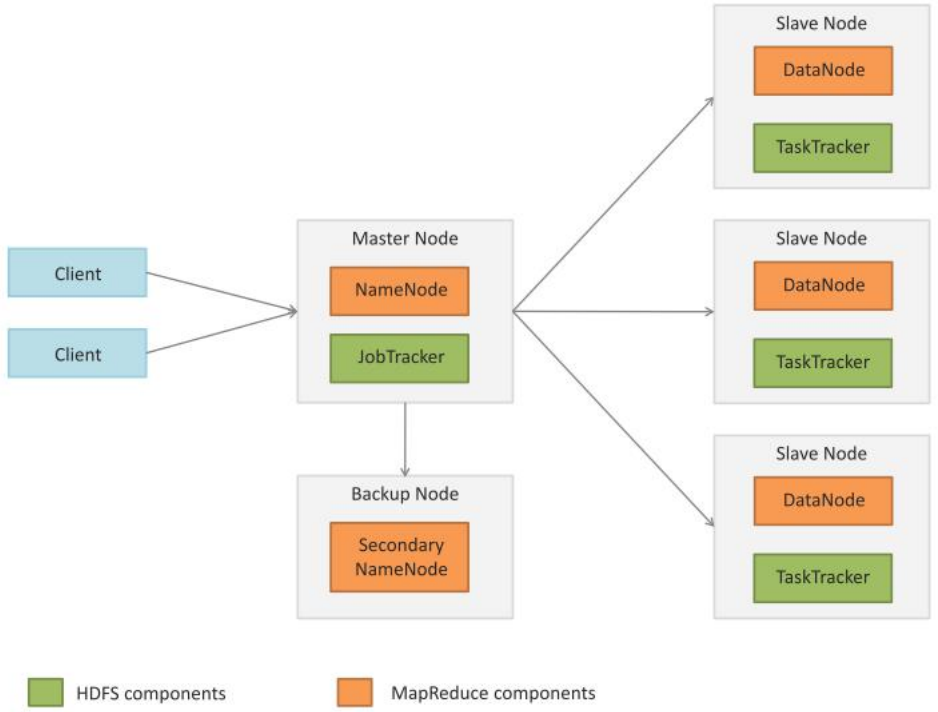
- A Hadoop cluster comprises of a Master node, backup node and a number of slave nodes.
- The master node runs the NameNode and JobTracker processes and the slave nodes run the DataNode and TaskTracker components of Hadoop.
- The backup node runs the Secondary NameNode process.
- NameNode
  - NameNode keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself. Client applications talk to the NameNode whenever they wish to locate a file, or when they want to add/copy/move/delete a file.
- Secondary NameNode
  - NameNode is a Single Point of Failure for the HDFS Cluster. An optional Secondary NameNode which is hosted on a separate machine creates checkpoints of the namespace.
- JobTracker
  - The JobTracker is the service within Hadoop that distributes MapReduce tasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack.



Apache Hadoop

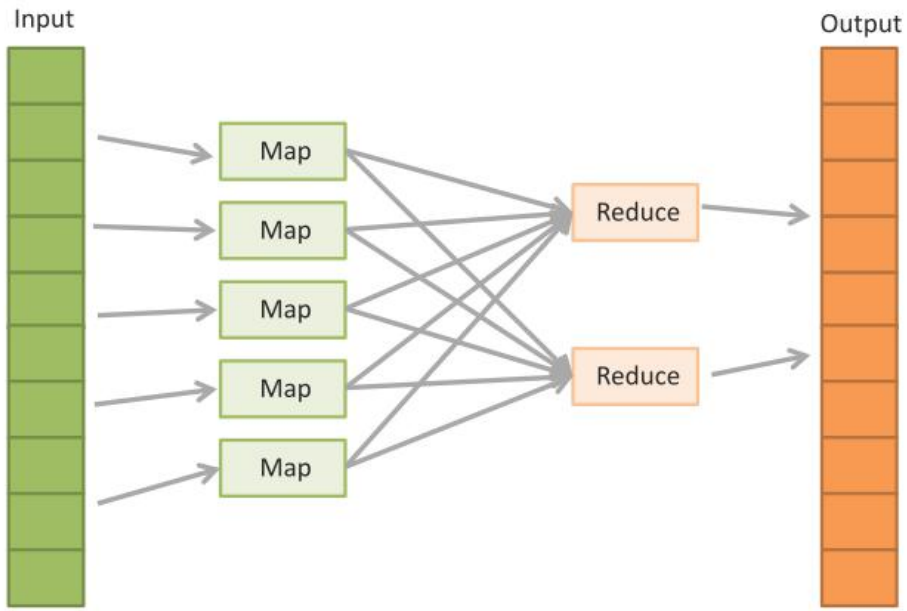
- TaskTracker
  - TaskTracker is a node in a Hadoop cluster that accepts Map, Reduce and Shuffie tasks from the JobTracker.
  - Each TaskTracker has a defined number of slots which indicate the number of tasks that it can accept.

- DataNode
  - A DataNode stores data in an HDFS file system.
  - A functional HDFS filesystem has more than one DataNode, with data replicated across them.
  - DataNodes respond to requests from the NameNode for filesystem operations.
  - Client applications can talk directly to a DataNode, once the NameNode has provided the location of the data.
  - Similarly, MapReduce operations assigned to TaskTracker instances near a DataNode, talk directly to the DataNode to access the files.
  - TaskTracker instances can be deployed on the same servers that host DataNode instances, so that MapReduce operations are performed close to the data.



MapReduce

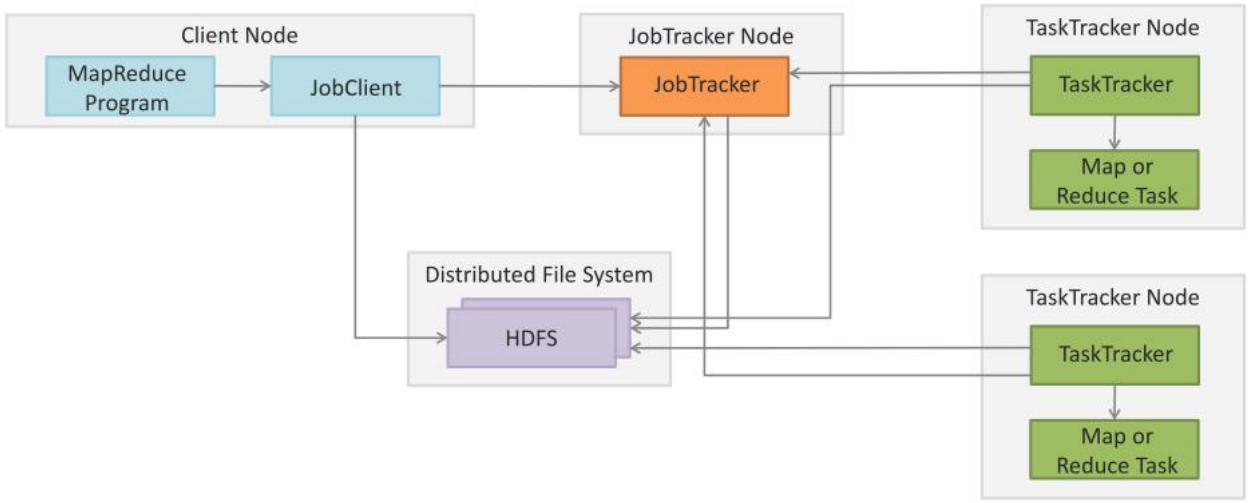
- MapReduce job consists of two phases:
  - Map: In the Map phase, data is read from a distributed file system and partitioned among a set of computing nodes in the cluster. The data is sent to the nodes as a set of key-value pairs. The Map tasks process the input records independently of each other and produce intermediate results as key-value pairs. The intermediate results are stored on the local disk of the node running the Map task.
  - Reduce: When all the Map tasks are completed, the Reduce phase begins in which the intermediate data with the same key is aggregated.



- Optional Combine Task
  - An optional Combine task can be used to perform data aggregation on the intermediate data of the same key for the output of the mapper before transferring the output to the Reduce task.

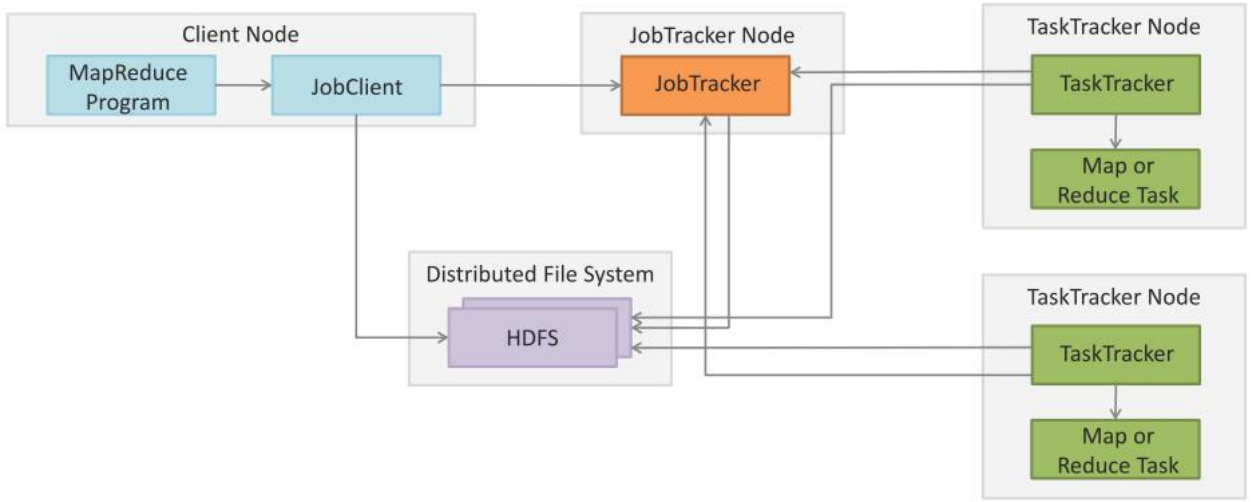
MapReduce Job Execution Workflow

- MapReduce job execution starts when the client applications submit jobs to the Job tracker.
- The JobTracker returns a JobID to the client application. The JobTracker talks to the NameNode to determine the location of the data.
- The JobTracker locates TaskTracker nodes with available slots at/or near the data.
- The TaskTrackers send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the JobTracker that they are still alive. These messages also inform the JobTracker of the number of available slots, so the JobTracker can stay up to date with where in the cluster, new work can be delegated.



MapReduce Job Execution Workflow

- The JobTracker submits the work to the TaskTracker nodes when they poll for tasks. To choose a task for a TaskTracker, the JobTracker uses various scheduling algorithms (default is FIFO).
- The TaskTracker nodes are monitored using the heartbeat signals that are sent by the TaskTrackers to JobTracker.
- The TaskTracker spawns a separate JVM process for each task so that any task failure does not bring down the TaskTracker.
- The TaskTracker monitors these spawned processes while capturing the output and exit codes. When the process finishes, successfully or not, the TaskTracker notifies the JobTracker. When the job is completed, the JobTracker updates its status.

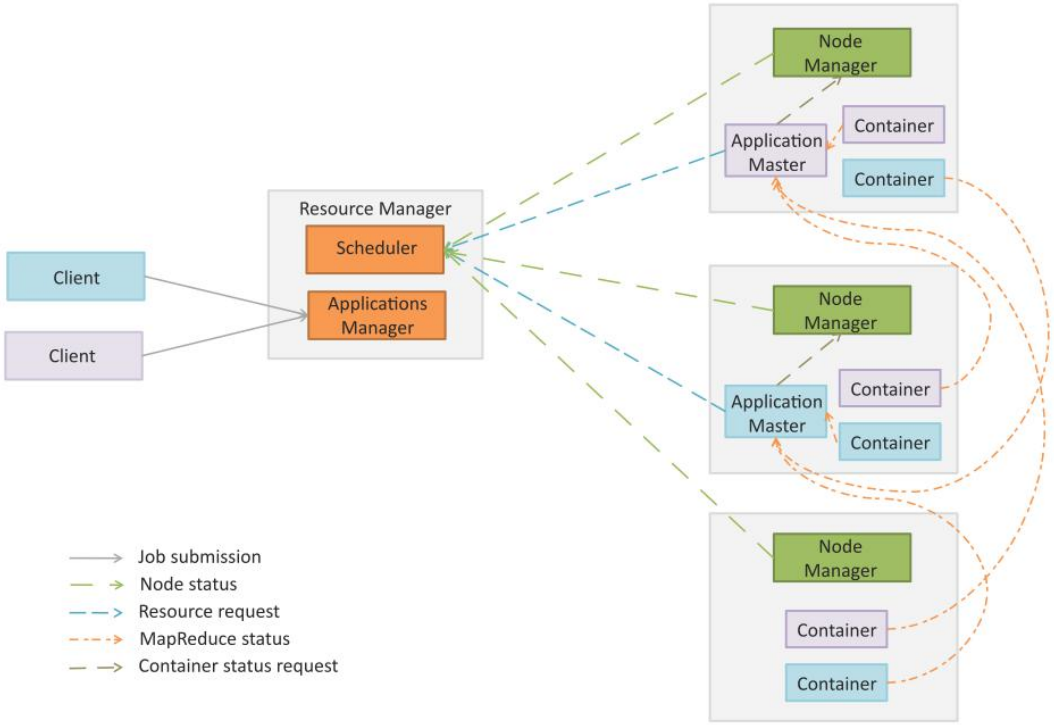


MapReduce 2.0 - YARN

- In Hadoop 2.0 the original processing engine of Hadoop (MapReduce) has been separated from the resource management (which is now part of YARN).

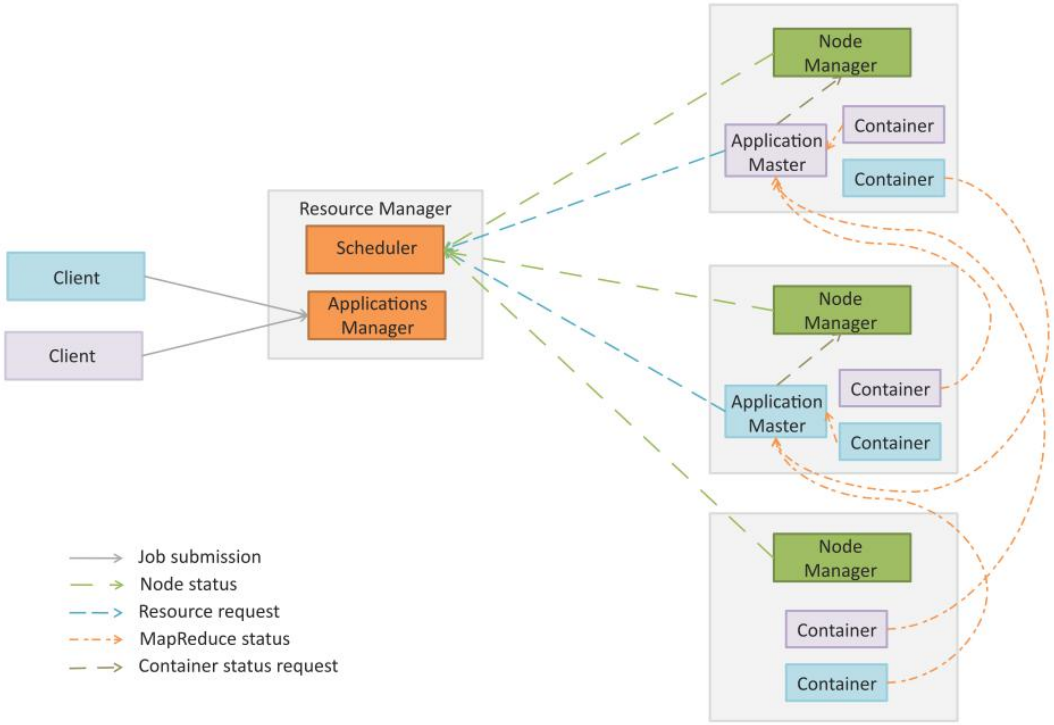
- This makes YARN effectively an operating system for Hadoop that supports different processing engines on a Hadoop cluster such as MapReduce for batch processing, Apache Tez for interactive queries, Apache Storm for stream processing, etc.

- YARN architecture divides architecture divides the two major functions of the JobTracker - resource management and job life-cycle management - into separate components:
  - ResourceManager
  - ApplicationMaster.



YARN Components

- **Resource Manager (RM)**: RM manages the global assignment of compute resources to applications. RM consists of two main services:
  - ***Scheduler***: Scheduler is a pluggable service that manages and enforces the resource scheduling policy in the cluster.
  - ***Applications Manager (AsM)***: AsM manages the running Application Masters in the cluster. AsM is responsible for starting application masters and for monitoring and restarting them on different nodes in case of failures.
- **Application Master (AM):** A per-application AM manages the application's life cycle. AM is responsible for negotiating resources from the RM and working with the NMs to execute and monitor the tasks.
- **Node Manager (NM):** A per-machine NM manages the user processes on that machine.
- **Containers:** Container is a bundle of resources allocated by RM (memory, CPU, network, etc.). A container is a conceptual entity that grants an application the privilege to use a certain amount of resources on a given machine to run a component task.



Hadoop Schedulers

- Hadoop scheduler is a pluggable component that makes it open to support different scheduling algorithms.
- The default scheduler in Hadoop is FIFO.
- Two advanced schedulers are also available - the Fair Scheduler, developed at Facebook, and the Capacity Scheduler, developed at Yahoo.
- The pluggable scheduler framework provides the flexibility to support a variety of workloads with varying priority and performance constraints.
- Efficient job scheduling makes Hadoop a multi-tasking system that can process multiple data sets for multiple jobs for multiple users simultaneously.

FIFO Scheduler

- FIFO is the default scheduler in Hadoop that maintains a work queue in which the jobs are queued.
- The scheduler pulls jobs in first in first out manner (oldest job first) for scheduling.
- There is no concept of priority or size of job in FIFO scheduler.

Fair Scheduler

- The Fair Scheduler allocates resources evenly between multiple jobs and also provides capacity guarantees.
- Fair Scheduler assigns resources to jobs such that each job gets an equal share of the available resources on average over time.
- Tasks slots that are free are assigned to the new jobs, so that each job gets roughly the same amount of CPU time.
- Job Pools
  - The Fair Scheduler maintains a set of pools into which jobs are placed. Each pool has a guaranteed capacity.
  - When there is a single job running, all the resources are assigned to that job. When there are multiple jobs in the pools, each pool gets at least as many task slots as guaranteed.
  - Each pool receives at least the minimum share.
  - When a pool does not require the guaranteed share the excess capacity is split between other jobs.
- Fairness
  - The scheduler computes periodically the difference between the computing time received by each job and the time it should have received in ideal scheduling.
  - The job which has the highest deficit of the compute time received is scheduled next.

Capacity Scheduler

- The Capacity Scheduler has similar functionally as the Fair Scheduler but adopts a different scheduling philosophy.
- Queues
  - In Capacity Scheduler, you define a number of named queues each with a configurable number of map and reduce slots.
  - Each queue is also assigned a guaranteed capacity.
  - The Capacity Scheduler gives each queue its capacity when it contains jobs, and shares any unused capacity between the queues. Within each queue FIFO scheduling with priority is used.
- Fairness
  - For fairness, it is possible to place a limit on the percentage of running tasks per user, so that users share a cluster equally.
  - A wait time for each queue can be configured. When a queue is not scheduled for more than the wait time, it can preempt tasks of other queues to get its fair share.

Hadoop Cluster Setup

Further Reading

- Apache Hadoop, http://hadoop.apache.org
- Apache Hive, http://hive.apache.org
- Apache HBase, http://hbase.apache.org
- Apache Chukwa, http://chukwa.apache.org
- Apache Flume, http://flume.apache.org
- Apache Zookeeper, http://zookeeper.apache.org
- Apache Avro, http://avro.apache.org
- Apache Oozie, http://oozie.apache.org
- Apache Storm, http://storm-project.net
- Apache Tez, http://tez.incubator.apache.org
- Apache Cassandra, http://cassandra.apache.org
- Apache Mahout, http://mahout.apache.org
- Apache Pig, http://pig.apache.org
- Apache Sqoop, http://sqoop.apache.org