ORACLE
Certified Professional

Java
Oracle Certified Professional Java Programmer

Microsoft Certified Professional

sohailimran@yahoo.com

APACHE hadoop Map Reduce

Sohail IMRAN سهيل عمران

intro

Ref: Web

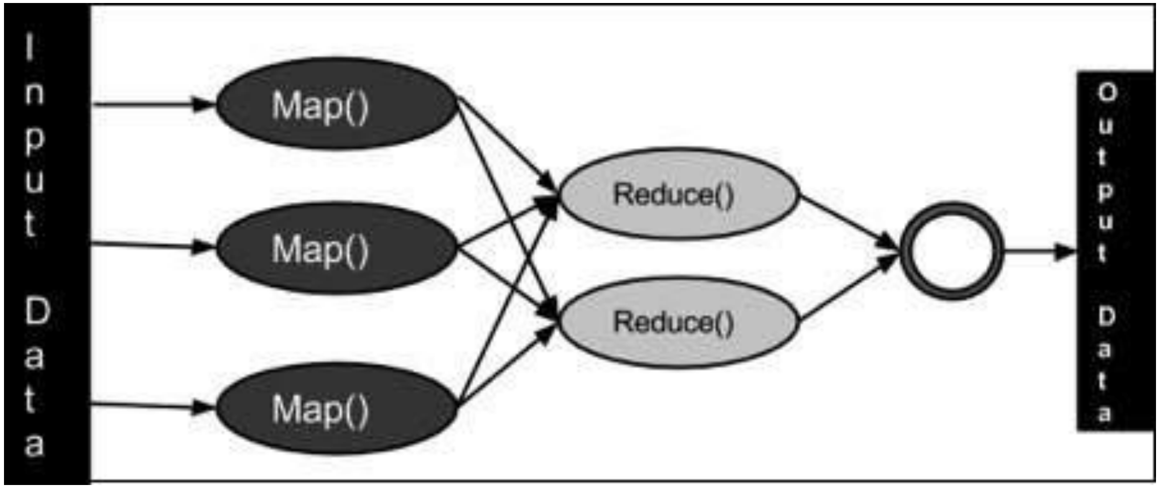
intro

MapReduce is a programming model or pattern within the Hadoop framework that is used to access big data stored in the Hadoop File System (HDFS).

It is a core component, integral to the functioning of the Hadoop framework.

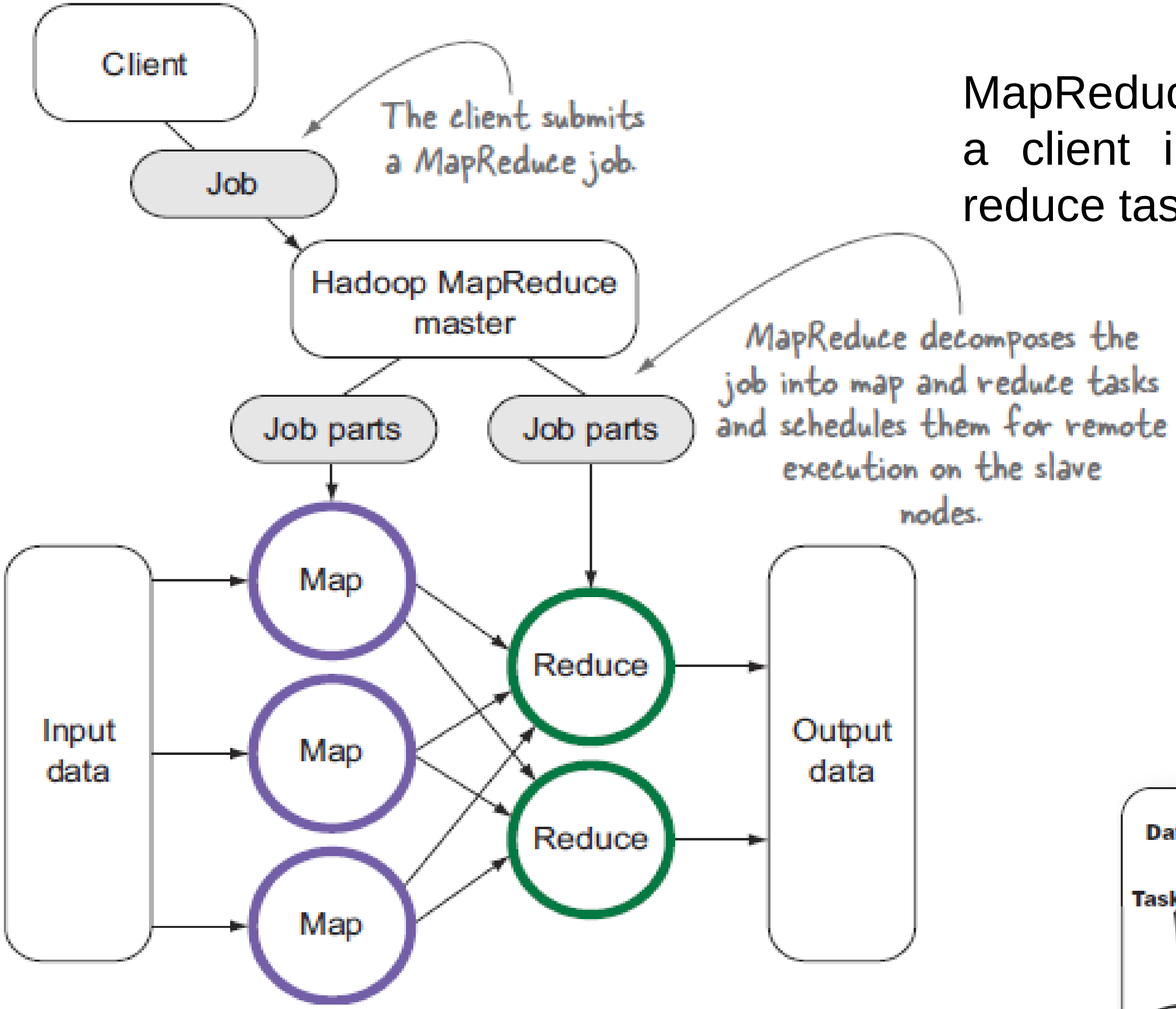
With MapReduce, rather than sending data to where the application or logic resides, the logic is executed on the server where the data already resides, to expedite processing.

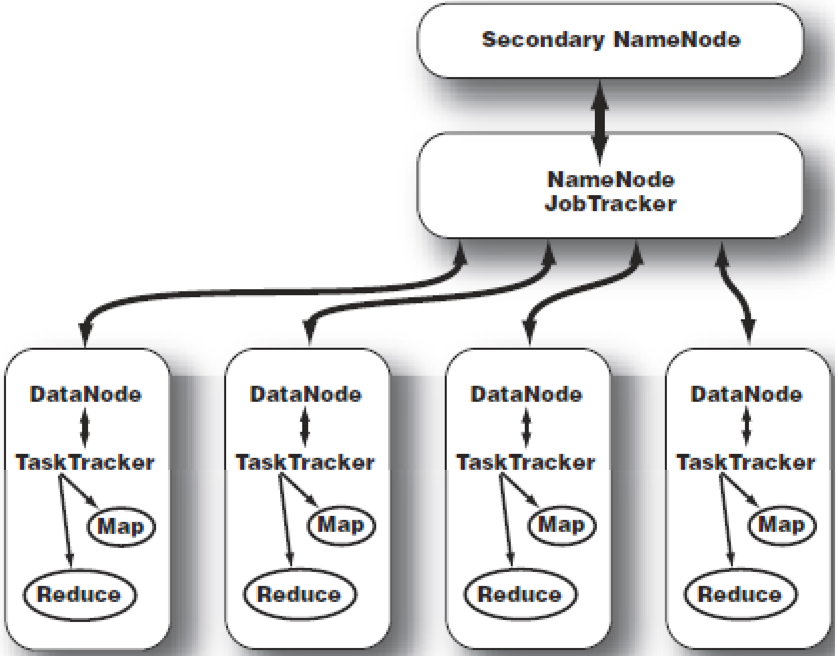
Data access and storage is disk-based—the input is usually stored as files containing structured, semi-structured, or unstructured data, and the output is also stored in files.



the mapreduce model

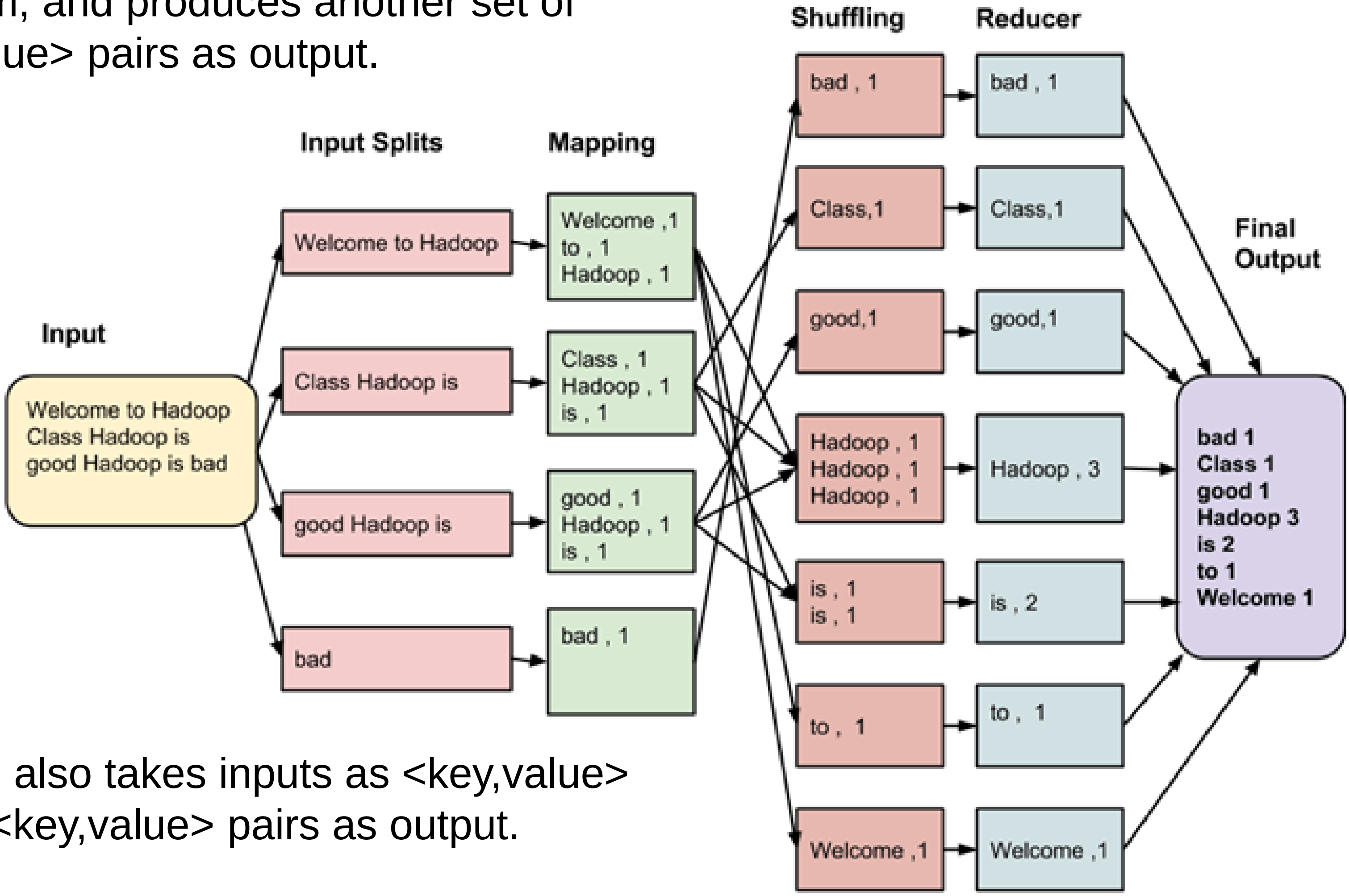
MapReduce decomposes work submitted by a client into small parallelized map and reduce tasks.


Client
The client submits a MapReduce job.
Job
Hadoop MapReduce master
MapReduce decomposes the job into map and reduce tasks and schedules them for remote execution on the slave nodes.
Job parts
Job parts
Input data
Map
Map
Map
Reduce
Reduce
Output data



example

The **Map** function takes input from the disk as <key,value> pairs, processes them, and produces another set of intermediate <key,value> pairs as output.



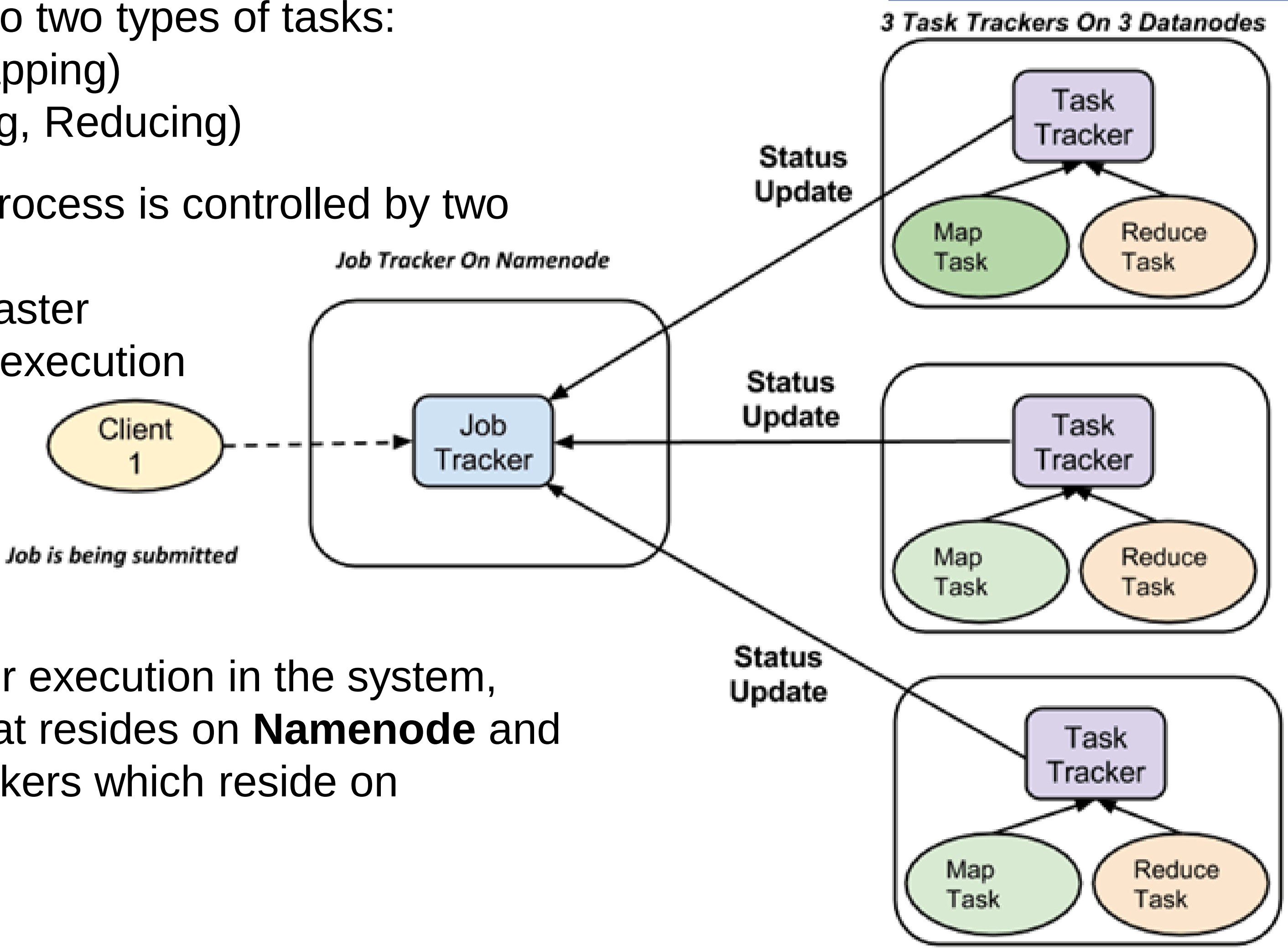
The **Reduce** function also takes inputs as <key,value> pairs, and produces <key,value> pairs as output.

architecture

Hadoop divides the job into two types of tasks:

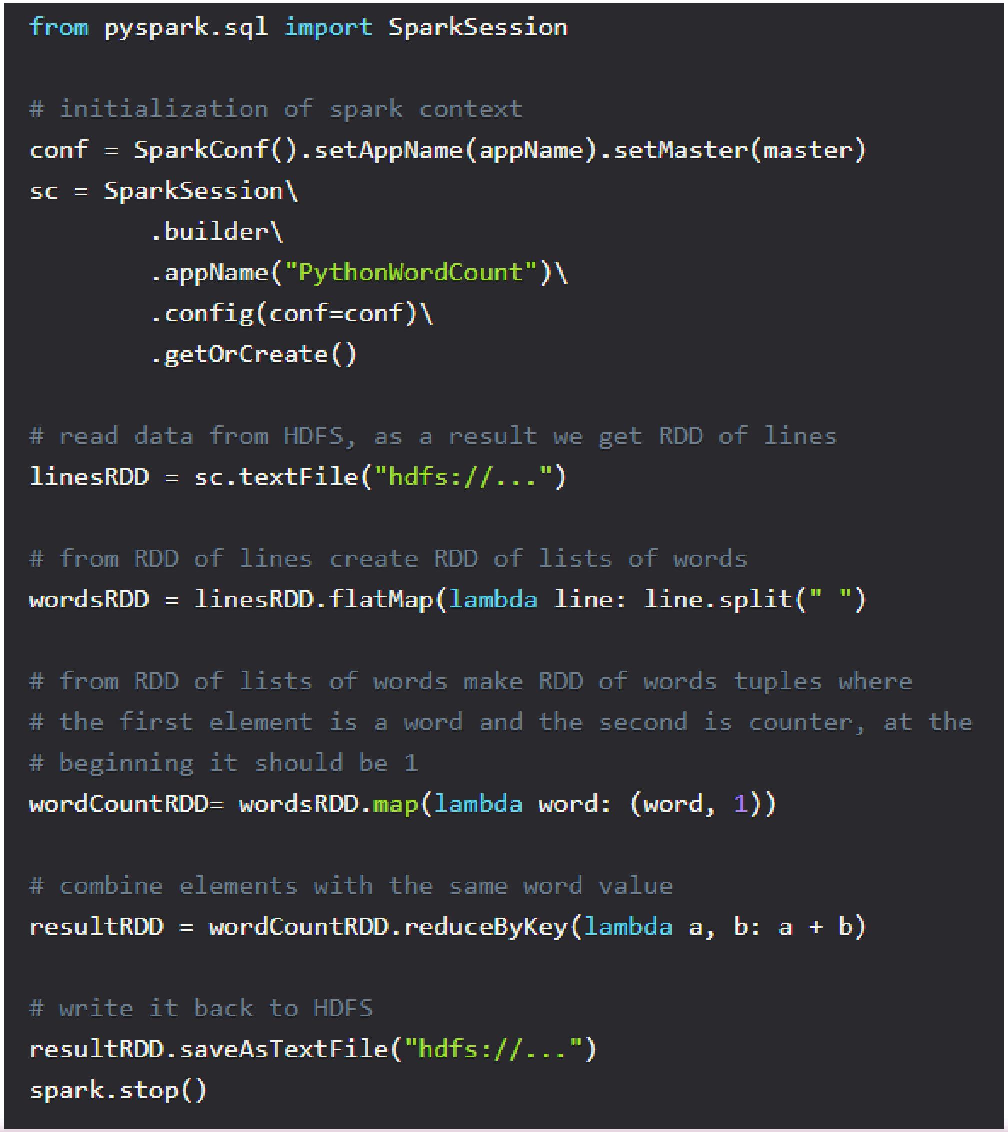
- **Map** tasks (Splits & Mapping)
- **Reduce** tasks (Shuffling, Reducing)

The complete execution process is controlled by two types of entities:
Jobtracker: Acts like a master (responsible for complete execution of submitted job)
Multiple Task Trackers: Acts like slaves, each of them performing the job



For every job submitted for execution in the system, there is one Jobtracker that resides on **Namenode** and there are multiple tasktrackers which reside on **Datanode**.

code



```
from pyspark.sql import SparkSession

# initialization of spark context
conf = SparkConf().setAppName(appName).setMaster(master)
sc = SparkSession\
        .builder\
        .appName("PythonWordCount")\
        .config(conf=conf)\
        .getOrCreate()

# read data from HDFS, as a result we get RDD of lines
linesRDD = sc.textFile("hdfs://...")

# from RDD of lines create RDD of lists of words
wordsRDD = linesRDD.flatMap(lambda line: line.split(" ")

# from RDD of lists of words make RDD of words tuples where
# the first element is a word and the second is counter, at the
# beginning it should be 1
wordCountRDD= wordsRDD.map(lambda word: (word, 1))

# combine elements with the same word value
resultRDD = wordCountRDD.reduceByKey(lambda a, b: a + b)

# write it back to HDFS
resultRDD.saveAsTextFile("hdfs://...")
spark.stop()
```