Introduction to Big Data

Big data is defined as collections of data sets whose volume, velocity in terms of time variation, or variety is so large that it is difficult to store, manage, process and analyze the data using traditional databases and data processing tools. In the recent years there has been an exponential growth in the both structured and unstructured data generated by information technology, industrial, healthcare, and other systems. Some examples of big data are described as follows:

- Data generated by social networks including text, images, audio and video data.
- Click-stream data generated by web applications such as e-Commerce to analyze user behavior.
- Machine sensor data collected from sensors embedded in industrial and energy systems for monitoring their health and detecting failures.
- Healthcare data collected in electronic health record (EHR) systems.
- Logs generated by web applications
- Stock markets data

The underlying characteristics of big data include:

- **Volume:** Though there is no fixed threshold for the volume of data to be considered as big data, however, typically, the term big data is used for massive scale data that is difficult to store, manage and process using traditional databases and data processing architectures. The volumes of data generated by modem IT, industrial, healthcare and systems is growing exponentially driven by the lowering costs of data storage and processing architectures and the need to extract valuable insights from the data to improve business processes, efficiency and service to consumers.
- **Velocity:** Velocity is another important characteristic of big data and the primary reason for exponential growth of data. Velocity of data refers to how fast the data is generated. Modem IT, industrial and other systems are generating data at increasingly higher speeds generating big data.
- **Variety:** Variety refers to the forms of the data. Big data comes in different forms such as structured or unstructured data, including text data, image, audio, video and sensor data.

Big data analytics involves several steps starting from data cleansing, data munging (or wrangling), data processing and visualization. Big data analytics is enabled by several technologies such as cloud computing, distributed parallel processing frameworks, in-memory databases, etc. In this chapter you will learn how to implement big data analytics including approaches for clustering and classification of big data.

Tiers of Big Data

Big Data is unstructured data that exceeds the processing complexity of conventional database systems. The data is too big, moves too fast, or doesn't fit the rule restricting behavior of our database architectures. This information comes from multiple, distinct, independent sources with complex and evolving relationships in a Big Data which is keep on growing day by day. There are three main challenges in Big Data which are data accessing and arithmetic computing procedures, semantics and domain knowledge for different Big Data applications and the difficulties raised by Big Data volumes, distributed data distribution and by complex and dynamic characteristics.

Big data framework is divided into three tiers as shown in figure 1, to handle the above challenges.

Tier I which is data accessing and computing focus on data accessing and arithmetic computing procedures. Because large amount of information are stored at different locations which are growing rapidly day by day, hence for computing distributed large-scale of information we have to consider effective computingplatform like Hadoop.

Figure 1. Three Tiers of Big Data Data privacy and domain knowledge is the Tier II which focuses on semantics and domain knowledge for different Big Data applications [9]. In social network, users are linked with each other that shares their knowledge which are represented by user communities, leaders in each group and social influence modelling and so on, therefore for understanding their semantics and application knowledge is important for both low-level data access and for high-level mining algorithm designs.

Tier III which is Big Data mining algorithm focus on difficulties raised by Big Data volumes, distributed data distribution and by complex and dynamic characteristics. There are three stages in Tier III, Sparse, heterogeneous, uncertain, incomplete and multisource data are pre-processed by data fusion techniques; (b) Complex and dynamic data are mined after pre-processing; (c) The global knowledge obtained by local learning and model fusion is tested and relevant information is feedback to the pre-processing stage

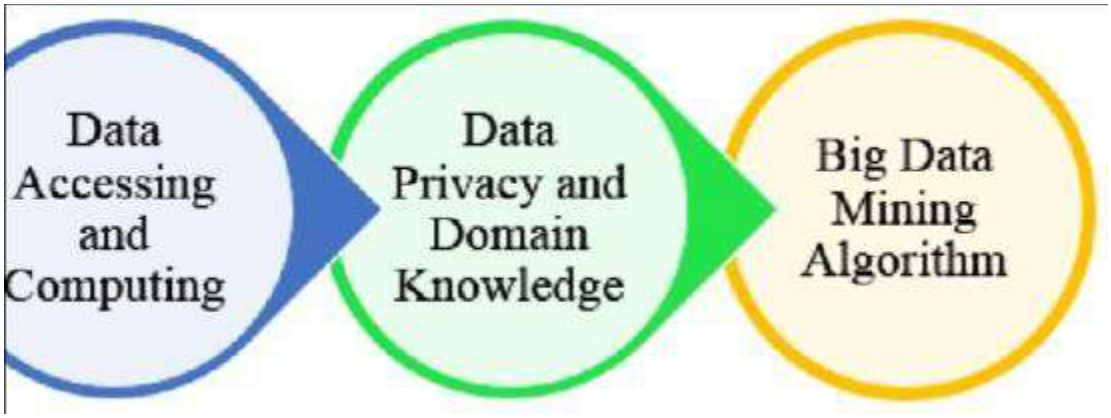


figure 1:- 3 tiers of big data

There are two main classification techniques, supervised and unsupervised. Supervised classification techniques , are also known as predictive or directed classification. In this method set of possible class is known in advanced. Unsupervised classification techniques are also known as descriptive or undirected. In this method set of possible class is unknown, after classification we can assign name to that class.

Classification of Big Data:

Classification technique is used to solve the above challenges which classify the big data according to the format of the data that must be processed, the type of analysis to be applied, the processing techniques at work, and the data sources for the data that the target system is required to acquire, load, process, analyze and store. Many classification techniques are used based on applications selected. Before actual classification begins, required information is extracted from large amount of data and then classification is done. There are two main classification techniques, supervised and unsupervised. Supervised classification techniques are also known as predictive or directed classification. In this method set of possible class is known in advanced. Unsupervised classification techniques are also known as descriptive or undirected. In this method set of possible class is unknown, after classification we can assign name to that class.

In supervised classification Decision Tree (DT) and Support Vector Machine (SVM) are well known classifier and used widely. Decision Tree is a hierarchical model that recursively does the separation of the input space into class regions. It consists of decision nodes and leaves. Learning algorithm for the Decision Tree is greedy, it finds the best attribute to split the data. Repeat this until it cannot be separated any more.

The main aim of DT is to find out the smallest tree that would make the data after split as pure as possible. Support Vector Machine is a supervised method that analyzes data and recognizes patterns which is used for classification. Given a training set and the data needs to be classified into two classes, a SVM classifier builds a model that assigns the data into one of the categories. Extraction of huge training set is modelled as a multi-dimensional classification problem with one class for each action and its aim is to assign a class label to a given action or activity.

2. OVERVIEW OF CLASSIFICATION

Classification is one of the data mining technique that classifies unstructured data into the structured class and groups and it helps to user for knowledge discovery and future plan . Classification provides intelligent decision making. There are two phases in classification, first is learning process phase in which a huge training data sets are

supplied and analysis takes place then rules and patterns are created. Then the execution of second phase start that is evaluation or test of data sets and archives the accuracy of a classification patterns. This section briefly describes the supervised classification methods such as Decision Tree and Support Vector Machine.

2.1 SUPERVISED METHODS

Problems which involve classification are considered to be instances of a branch of machine learning called as “supervised learning” [5]. In this, the machine is given a “training set” of correctly classified instances of data in the first stage, and then the algorithm devised from this “learning” is used for the next stage of prediction. The converse of this is “unsupervised learning”, which involves classifying data into categories based on some similarity of input parameters in the data.

Supervised Data Mining techniques are appropriate when we have a specific target value, so we can predict about our data [8]. The targets can have two or more possible outcomes, or even be a continuous numeric value. To use these methods, we ideally have a subset of training data (observations and measurement) sets for which this target value is already known. Training data includes both the input and desired results. New data is classified based on the training set. The input data also called the training set which consists of multiple attributes or features. Each record is tagged with a class label. For some examples the correct results (target) are known and are given in input to the model during the learning process. The construction of a proper training validation and test set is crucial. These methods are usually fast and accurate. The objective of classification is to analyze huge data and to develop an accurate description or model for each organized class using the feature present in the data. We use that training data to build a model of what a typical data set looks like when it has one of the various target values. We then apply that model to data for which that target value is currently unknown. The algorithm identifies the “new” data points that match the model of each target value. This model is used to classify test data for which the class descriptions are not known.

1. Decision Tree (DT)

Decision Tree is ideal to use as the filter to handle the large amount of data. DT is a basic way of classification can have satisfactory efficiency and accuracy of those datasets.

Decision Tree algorithm is good at tuning between precision which can be trained very fast and provide sound results on those classification data [2].

Big Data are now rapidly expanding in all domains with the fast development of networking and increase in the data storage and collection capacity. The instances are divided into a set of discrete valued set of properties, known as various features of the data. For example, classifying a received email as “spam” or “not spam” could be based on analyzing characteristics of the email such as origin IP address, the number of emails received from the same origin, the subject line, the email address itself, the content of the body of the email, etc. All these features will contribute to a final value which will allow the algorithm to classify the email. It is logical that the more number of examples of spam and non-spam emails the Machine Learning system goes through, the better will be its prediction for the next unknown email.

Decision Tree learning is reasonably fast and accurate. The approach is to learn on large data sets is to parallelize the process of learning by utilizing Decision Trees. It is straightforward to reduce a Decision Tree to rules. The strategy follow here is to break a large data set into n partitions then learn a DT on each of the n partitions in parallel. A DT become bigger on each of n processors independently. After that they must be combined in such a way that the Decision Tree remains individual tree, for this approach Decision

Tree can used Meta-learning. Meta-learning is the process by which learners become aware of and increasingly in control of habits of perception, inquiry, learning, and growth.

Now other aspect of creating final DT is pruning the tree which removes the nodes that do not provides accuracy in classification results in reduced size tree. Pruning is likely to be very important for large

training set which will produce large trees. There are a number of methods to prune a Decision Tree. In

C4.5 an approach called pessimistic pruning is quite fast and has been shown to provide trees that perform

adequately.

Big Data challenges are growing day by day, traditional Decision Tree algorithms have multiple limitations.

First, building a Decision Tree is a very time consuming when the available dataset is

extremely huge.Second, although parallel computing clusters can be leveraged in Decision Tree based classification algorithms, the strategy of data distribution should be optimized so that required data for building one node is localized and meanwhile the communication cost is minimized. To overcome these limitations, distributed C4.5 algorithm with MapReduce computing model is used.

When available dataset is extremely huge then C4.5 algorithm performs well in short time and it is robust in nature as well as simple to understand [10].

2. Support Vector Machine (SVM)

Machine Learning has ability to enable the computer to learn that uses algorithm and techniques which perform different tasks and activities to provide efficient learning. Our main problem is that how can we represent complex data and how to exclude bogus data. Support Vector Machine is a Machine Learning tool used for classification that is based on Supervised Learning which classifies points to one of two disjoint half-spaces. It uses nonlinear mapping to convert the original data into higher dimension. Its objective is to construct a function which will correctly predict the class to which the new point belongs and the old points belong. In the era of Big Data, the main reason behind maximum margin or separation because if we use a decision boundary to classify, it may end up nearer to one set of datasets compared to others. This happens only if data is structured or linear but mostly we find data is unstructured/nonlinear and dataset is inseparable then SVM kernels are used.

Traditional Classification approaches perform weakly when working directly because of huge amount of data but Support Vector Machine can avoid the problems of representing this much data. Support Vector Machine is the most promising technique and approach as compared to others classification approaches.

Support Vector Machine balance proper and accurate huge amount of data and compromise between classifier complexity and error can be controlled explicitly. Another benefit of SVMs is that one can design and use a SVM kernel for a particular problem that could be applied directly to the data without the need for a feature extraction process. It is particularly important problems, where huge amount of structured data is lost by the feature extraction process.

Support Vector Machine (SVM) is the classification technique which used to process on large training data.

The Big and complex data can be left to the SVM since the result of SVM will be greatly influenced when there is too much noise in the datasets. SVM provides with an optimized algorithm to solve the problem of over fitting. SVM is an effective classification model is useful to handle those complex data. SVM can make use of certain kernels to reveal efficiently in quantum form the largest eigenvalues and corresponding eigenvectors of the training data overlap (kernel) and covariance matrices [2].

SVM have high training performance and low generalization error which pointed out the potential problems of SVMs when the training set is noisy and imbalanced. The SVM is not that much scalable on large data sets because it take time for multiple scanning of data sets hence it is too expensive to perform.

To overcome this problem, Clustering-Based SVM (CB-SVM) comes into picture for scalability and reliability of SVM classification [6]. Clustering-Based SVM (CB-SVM) is the SVM technique that is designed for handling large data sets which applies on hierarchical micro-clustering algorithm that scans the entire data set only once to provide the high quality of samples. CB-SVM is scalable if and only if the efficiency of training maximizing the performance of SVMs.

Classification is the process of categorizing objects into predefined categories. Classification is achieved by classification algorithms that belong to a broad category of algorithms called supervised machine learning. Supervised learning involves inferring a model from a set of input data and known responses to the data (training data) and then using the inferred model to predict responses to new data. There are various types of classification approaches for big data analytics including:

- **Binary classification:** Binary classification involves categorizing the data into two categories. For example, classifying the sentiment of a news article into positive or negative, classifying the state of a machine into good or faulty, classifying the heath test into positive or negative, etc.
- **Multi-class classification:** Multi-class classification involves more than two classes into which the data is categorized. For example, gene expression classification problem involves multiple classes.
- **Document classification:** Document classification is a type of multi-class classification approach in which the data to the classified is in the form of text document. For classifying news articles into different categories such as politics, sports, etc.

Introduction to Hadoop, Hadoop Map Reduce Job Execution

Apache Hadoop

Apache Hadoop [102] is an open source framework for distributed batch processing of big data. MapReduce has also been proposed as a parallel programming model [14] suitable for the cloud. The MapReduce algorithm allowed large scale computations to be parallelized across a large cluster of servers.

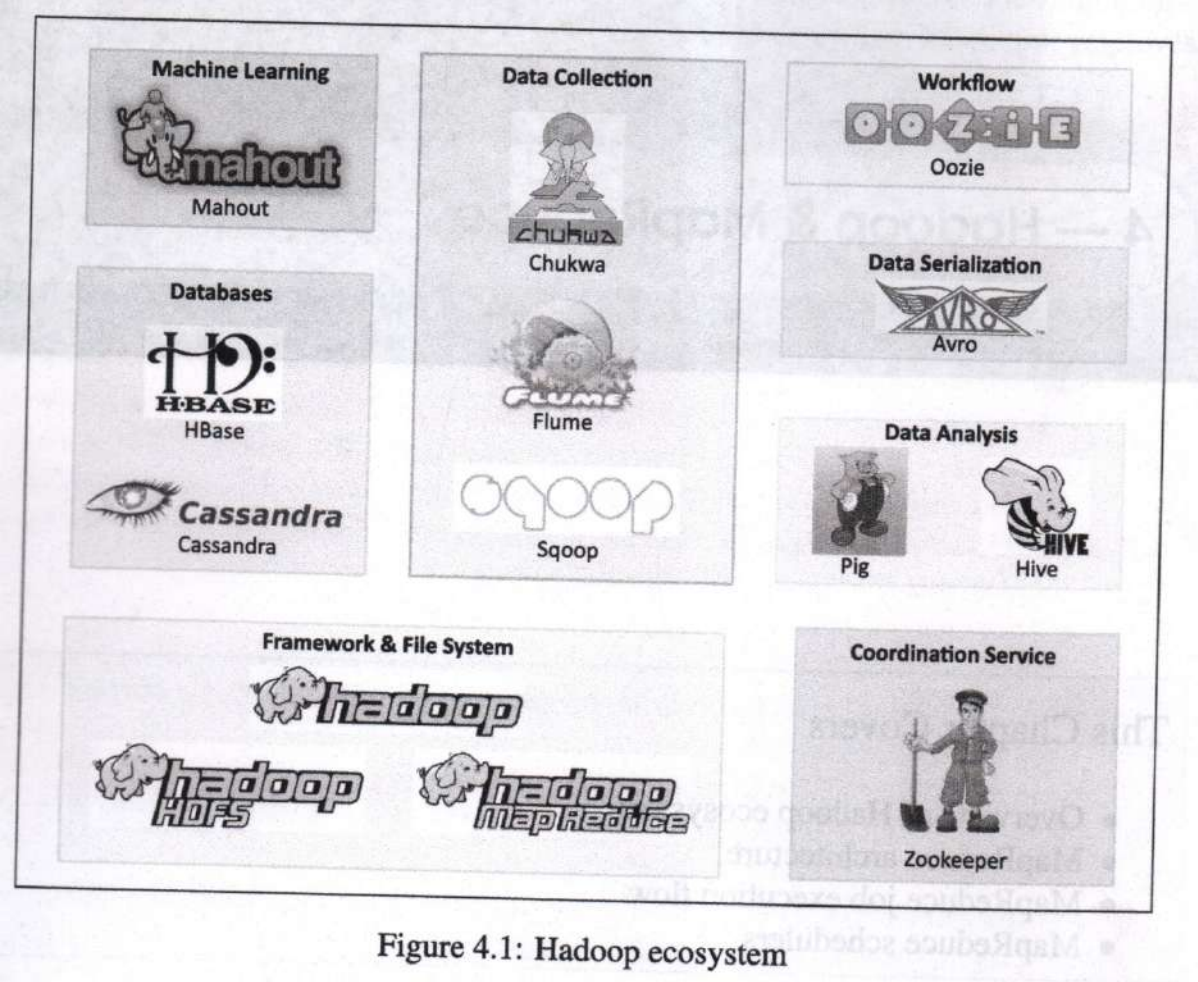


Figure 4.1: Hadoop ecosystem

The Hadoop ecosystem consist of a number of projects as shown in Figure 4.1. These projects are described briefly as follows:

- **Hadoop Common:** Hadoop Common consists of common utilities that support other Hadoop modules. Hadoop common has utilities and scripts

for starting Hadoop, components and interfaces to access the file systems supported by Hadoop.

- **Hadoop Distributed File System (HDFS):** HDFS is a distributed file system (DFS) that runs on large clusters and provides high throughput access to data. HDFS was built to reliably store very large files across machines in a large cluster built of commodity hardware. HDFS stores each file as a sequence of blocks all of which are of the same size except the last block. The blocks of each file are replicated on multiple machines in a cluster with a default replication factor of 3 to provide fault tolerance.

- **Hadoop MapReduce:** MapReduce is a parallel programming model that allows distributing large scale computations in a sequence of operations on data sets of key-value pairs. The Hadoop MapReduce provides a data processing model and an execution environment for MapReduce jobs for large scale data processing.

- **Hadoop YARN:** A framework for job scheduling and cluster resource management.

- **HBase:** HBase is a scalable, non-relational, distributed, column-oriented database that provides structured data storage for large tables [19]. HBase data storage can scale linearly and automatically by addition of new nodes. As the tables grow, they are automatically split into regions and distributed across all available datanodes.

- **Zookeeper:** Zookeeper is a high performance distributed coordination service for maintaining configuration information, naming, providing distributed synchronization and group services [22].

- **Pig:** Pig is a data flow language and an execution environment for analyzing large datasets. Pig compiler produces a sequence of MapReduce jobs that analyze data in HDFS using the Hadoop MapReduce framework [29],

- **Hive:** Hive is a distributed data warehouse infrastructure for Hadoop. Hive provides an SQL-like language called HiveQL that allows easy data summarization, ad-hoc querying, and analysis of large datasets stored in HDFS [18].

- **Chukwa:** Chukwa is a data collection system for monitoring large distributed systems. Chukwa is built on top of HDFS and Hadoop MapReduce and allows collecting and analyzing data [20],

- **Mahout:** Mahout is a scalable machine learning library for Hadoop. Mahout provides a large collection of machine learning algorithms for clustering, classification and collaborative filtering which are implemented on top of Hadoop using the MapReduce parallel programming model [68].
- **Cassandra:** Cassandra is scalable multi-master database with no single points of failure. Cassandra is designed to handle massive scale data spread across many servers and provides a highly available service with no single point of failure. Cassandra is a No-SQL solution that provides a structured key-value store [27].
- **Avro:** Avro is a data serialization system that provides rich data structures, a compact and fast binary data format, a container file to store persistent data, cross-language RPC and simple integration with dynamic languages [23].
- **Oozie:** Oozie is a workflow scheduler system for managing Hadoop jobs. Oozie is integrated with the Hadoop stack and supports several types of Hadoop jobs such as MapReduce jobs, Pig, Hive, Sqoop, Distcp and system specific jobs [24].
- **Flume:** Flume is a distributed, reliable and available service for collecting, analyzing and moving large amounts of data from applications to HDFS [21].
- **Sqoop:** Sqoop is a tool that allows efficiently transferring bulk data between Hadoop and structured datastores such as relational databases [30].

4.1 Hadoop MapReduce Job Execution

A MapReduce job consists of two phases. 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. When all the Map tasks are completed, the Reduce phase begins in which the intermediate data with the same key is aggregated.

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. In this section you will learn about the MapReduce job execution workflow and the steps involved in job submission, job initialization, task selection and task execution.

Figure 4.2 shows the components of a Hadoop cluster. 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.

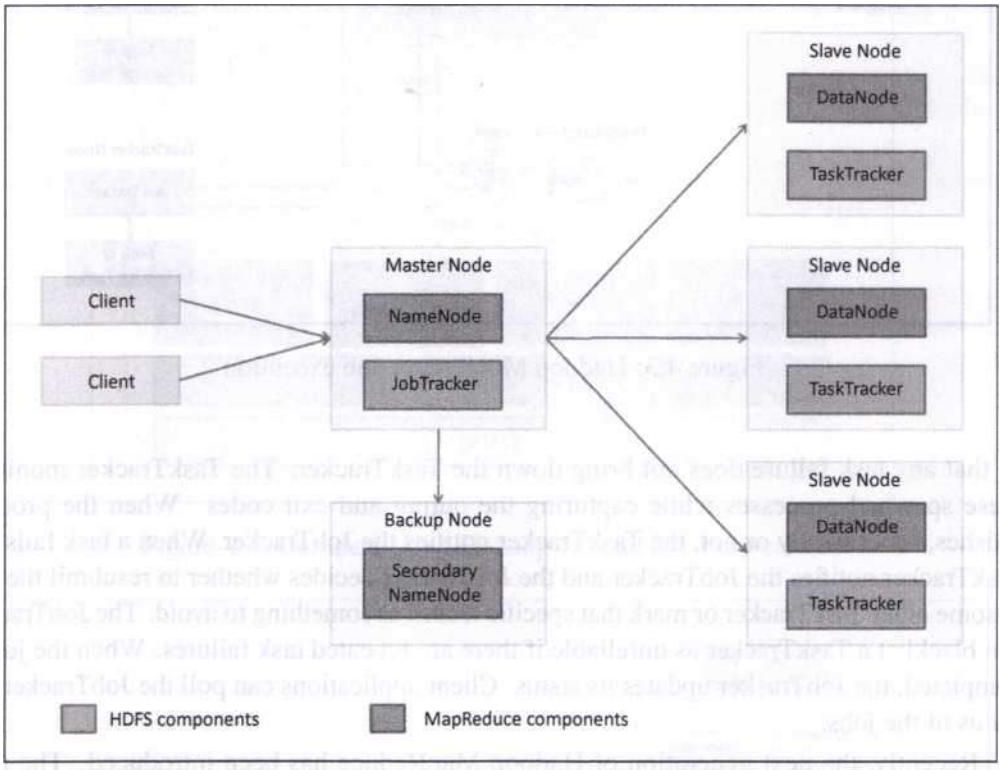


Figure 4.2: Components of a Hadoop cluster

The functions of the key processes of Hadoop are described as follows:

4.2.1 NameNode

NameNode keeps the directory4cee 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. The NameNode responds to the successful requests by returning a list of relevant DataNode servers where the data lives. NameNode serves as both directory namespace manager and ’inode table’ for the Hadoop DFS. There is a single NameNode running in any DFS deployment.

4.2.2 Secondary NameNode

HDFS is not currently a high availability system. The NameNode is a Single Point of Failure for the HDFS Cluster. When the NameNode goes down, the file system goes offline. An optional Secondary NameNode which is hosted on a separate machine creates checkpoints of the namespace.

4.2.3 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.

4.2.4 TaskTracker

TaskTracker is a node in a Hadoop cluster that accepts Map, Reduce and Shuffle tasks from the JobTracker. Each TaskTracker has a defined number of slots which indicate the number of tasks that it can accept. When the JobTracker tries to find a TaskTracker to schedule a map or reduce task it first looks for an empty slot on the same node that hosts the DataNode containing the data. If an empty slot is not found on the same node, then the JobTracker it looks for an empty slot on a node in the same rack.

4.2.5 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 connect to the NameNode on startup. 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 Job Execution Workflow

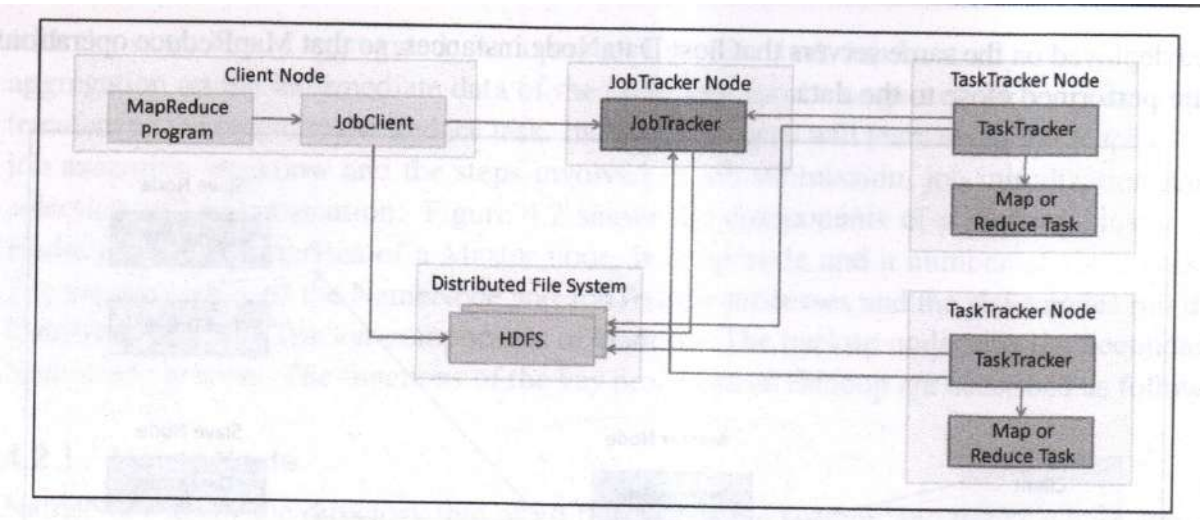


Figure 4.3: Hadoop MapReduce job execution

Figure 4.3 shows the MapReduce job execution workflow for first generation Hadoop MapReduce framework (MR1). The 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. 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.

The default scheduling algorithm in Hadoop is FIFO (first-in, first-out). In FIFO scheduling a work queue is maintained and JobTracker pulls the oldest job first for scheduling. There is no notion of the job priority or size of the job in FIFO scheduling.

Hadoop, however, implements the ability for pluggable schedulers. There are other advanced scheduling algorithms that come with Hadoop.

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 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 a task fails the TaskTracker notifies the JobTracker and the JobTracker decides whether to resubmit the job to some other TaskTracker or mark that specific record as something to avoid.

The JobTracker can blacklist a TaskTracker as unreliable if there are repeated task failures. When the job is completed, the JobTracker updates its status. Client applications can poll the JobTracker for status of the jobs.

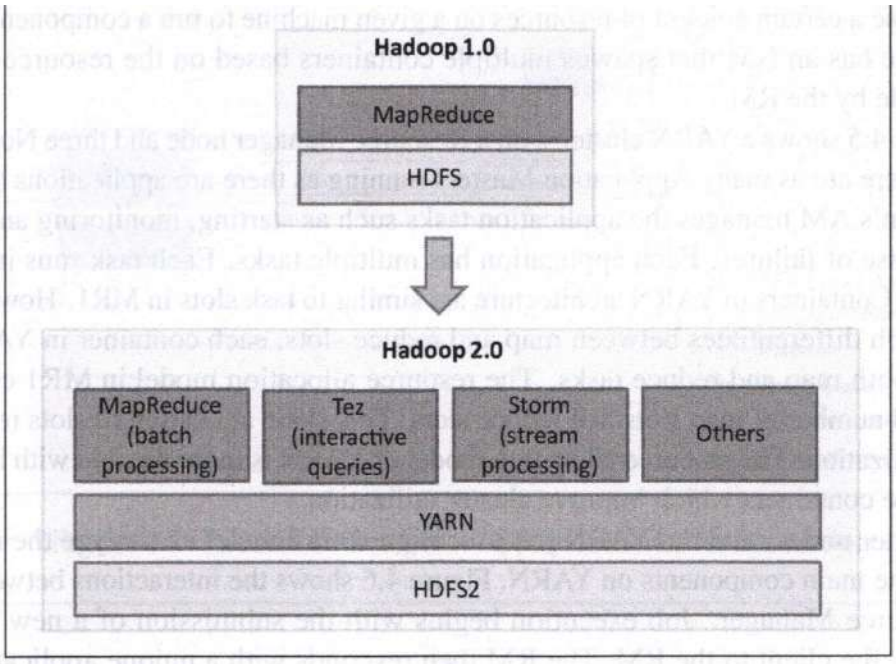


Figure 4.4: Comparison of Hadoop 1.0 and 2.0 architectures

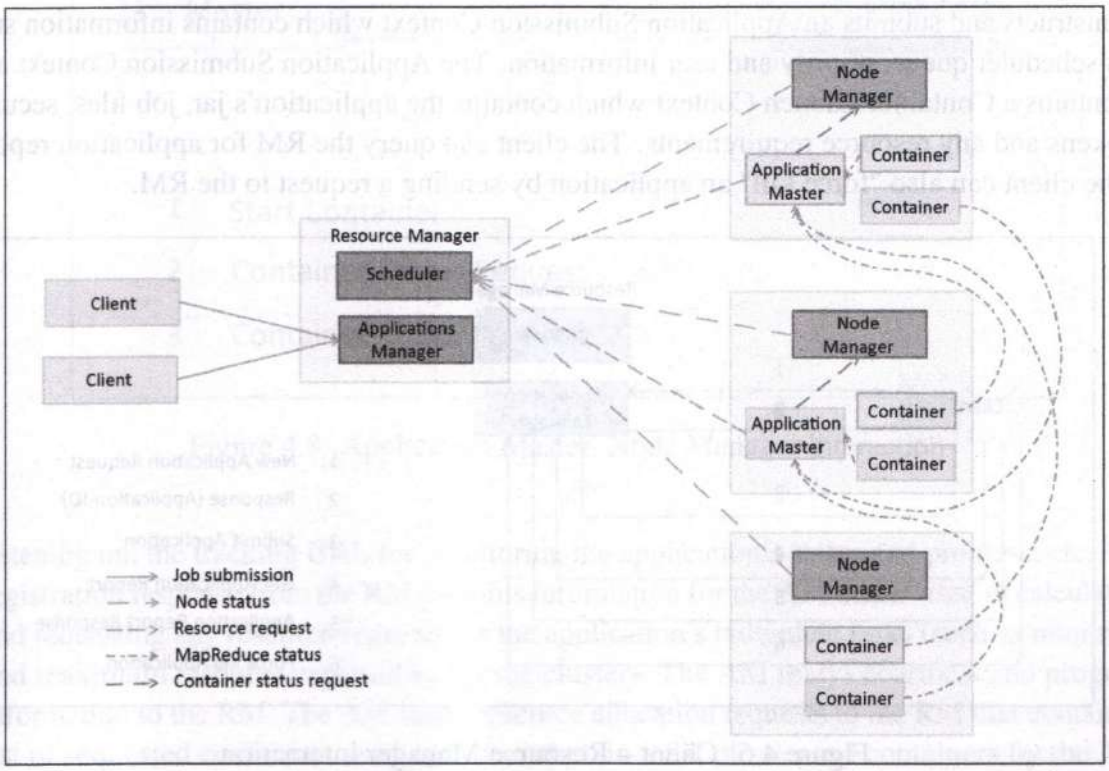


Figure 4.5: Hadoop MapReduce Next Generation (YARN) job execution

Recently, the next generation of Hadoop MapReduce has been introduced. The new MapReduce architecture is called YARN or MapReduce 2.0 (MR2). In Hadoop 2.0 the original processing engine of Hadoop (MapReduce) has been separated from the resource management (which is now part of YARN) as shown in Figure 4.4. 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 [26] for interactive queries, Apache Storm [25] for stream processing, etc.

Figure 4.5 shows the MapReduce job execution workflow for next generation Hadoop MapReduce framework (MR2). The next generation MapReduce architecture divides the two major functions of the JobTracker - resource management and job life-cycle management - into separate components - ResourceManager and ApplicationMaster. The key components of YARN are described as follows:

- **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 task

to use a certain amount of resources on a given machine to run a component task. Each node has an NM that spawns multiple containers based on the resource allocations made by the RM.

Figure 4.5 shows a YARN cluster with a Resource Manager node and three Node Manager nodes. There are as many Application Masters running as there are applications (jobs).

Each application's AM manages the application tasks such as starting, monitoring and restarting tasks in case of failures.

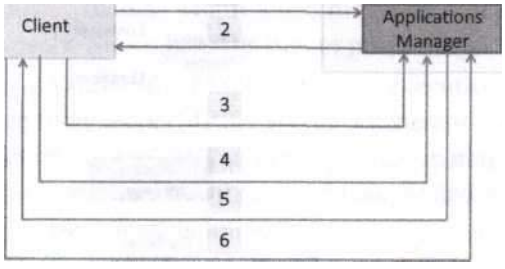
Each application has multiple tasks. Each task runs in a separate container. Containers in YARN architecture are similar to task slots in MR1.

However, unlike MR1 which differentiates between map and reduce slots, each container in YARN can be used for both map and reduce tasks. The resource

allocation model in MR1 consists of a predefined number of map slots and reduce slots. This static allocation of slots results in low cluster utilization.

The resource allocation model of YARN is more flexible with introduction of resource containers which improve cluster utilization.

To better understand the YARN job execution workflow let us analyze the interactions between the main components on YARN. Figure 4.6 shows the interactions between a Client and Resource Manager. Job execution begins with the submission of a new application request by the client to the RM. The RM then responds with a unique application ID and information about cluster resource capabilities that the client will need in requesting resources for running the application's AM. Using the information received from the RM, the client constructs and submits an Application Submission Context which contains information such as scheduler queue, priority and user information. The Application Submission Context also contains a Container Launch Context which contains the application's jar, job files, security tokens and any resource requirements. The client can query the RM for application reports. The client can also "force kill" an application by sending a request to the RM.



Resource Manager

Scheduler

1 New Application Request

2 Response (Application-ID)

3 Submit Application

4 Get Application Report

5 Application Report Response

6 Force Kill Application

Figure 4.6: Client - Resource Manager interaction

Figure 4.7 shows the interactions between Resource Manager and Application Master. Upon receiving an application submission context from a client, the RM finds an available container meeting the resource requirements for running the AM for the application. On finding a suitable container, the RM contacts the NM for the container to start the AM process on its node. When the AM is launched it registers itself with the RM. The registration process consists of handshaking that conveys information such as the RPC port that the AM will be listening on, the tracking URL for monitoring the application's status and progress, etc. The registration response from the RM contains information for the AM that is used in calculating and requesting any resource requests for the application's individual tasks (such as minimum and maximum resource capabilities for the cluster). The AM relays heartbeat and progress information to the RM. The AM sends resource allocation requests to the RM that contains a list of requested containers, and may also contain a list of released

containers by the AM. Upon receiving the allocation request, the scheduler component of the RM computes a list of containers that satisfy the request and sends back an allocation response. Upon receiving the resource list, the AM contacts the associated NMs for starting the containers. When the job finishes, the AM sends a Finish Application message to the RM.

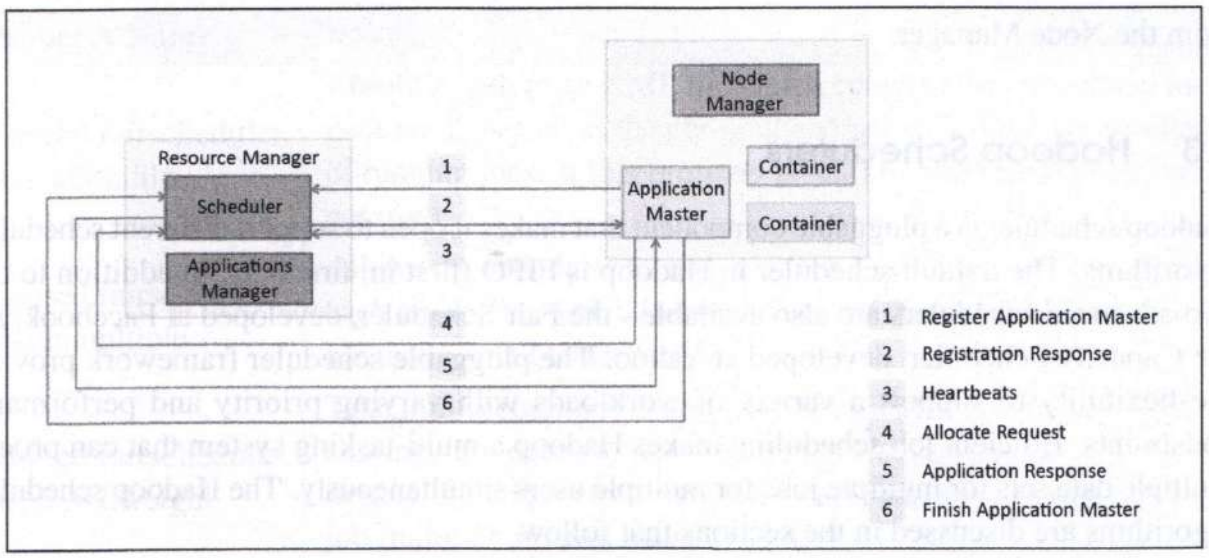


Figure 4.7: Resource Manager – Application Master interaction

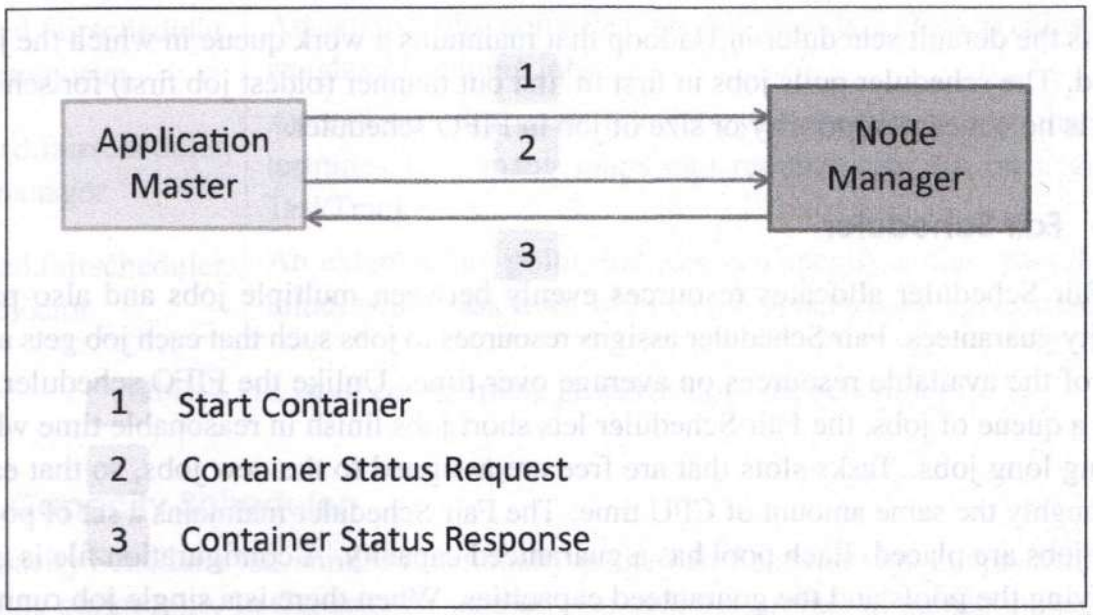


Figure 4.8: Application Master- Node Manager interaction

Figure 4.8 shows the interactions between the an Application Master and Node Manager. Based on the resource list received from the RM, the AM requests the hosting NM for each container to start the container. The AM can request and receive a container status report from the Node Manager.

Hadoop Schedulers

Hadoop scheduler is a pluggable component that makes it open to support different scheduling algorithms. The default scheduler in Hadoop is FIFO (first in, first out).

In addition to this, 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.

The Hadoop scheduling algorithms are discussed in the sections that follow.

4.3.1 FIFO

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.

4.3.2 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. Unlike the FIFO scheduler, which forms a queue of jobs, the Fair Scheduler lets short jobs finish in reasonable time while not starving long jobs.

Tasks slots that are free are assigned to the new jobs, so that each job gets roughly the same amount of CPU time. The Fair Scheduler maintains a set of pools into which jobs are placed.

Each pool has a guaranteed capacity. A configuration file is used for specifying the pools and the guaranteed capacities. 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. This lets the scheduler guarantee capacity for pools while utilizing resources efficiently when these pools don’t contain jobs.

All the pools have equal share by default. It is possible to provide more or less share to a pool by specifying the share in the configuration file. The Fair Scheduler keeps track of the compute time received by each job. 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. This ensures that over time, each job gets its fair share of compute time. When multiple users are submitting jobs, to ensure fairness, each user is assigned to a pool. It is possible to limit the number of running jobs per user or per pool by specifying the limits

Property Name	Details
mapred.fairscheduler.allocation.file	Absolute path to an XML file which contains the allocations for each pool, as well as the per-pool and per-user limits on number of running jobs. If this property is not provided, allocations are not used.
mapred.fairscheduler.assignmultiple	Allows the scheduler to assign a map task and a reduce task on each heartbeat, which improves cluster throughput when there are many small tasks to run.
mapred.fairscheduler.sizebasedweight	Take into account job sizes in calculating their weights for fair sharing. By default, weights are only based on job priorities. Setting this flag to true will make them based on the size of the job (number of tasks needed) as well.
mapred.fairscheduler.poolnameproperty	Specify which jobconf property is used to determine the pool that a job belongs in.
mapred.fairscheduler.	An extensibility point that lets you specify a class

in the configuration file. This is useful when a user submits a large number of jobs and permits greater responsiveness of the Hadoop cluster. Table 4.1 lists the configurable properties of the Fair Scheduler

Table 4.1: List of configurable properties of Fair Scheduler

4.3.3 Capacity Scheduler

The Capacity Scheduler has similar functionally as the Fair Scheduler but adopts a different scheduling philosophy. 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. 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.

Capacity Scheduler allows enforcing strict access control on queues. The access controls are defined on a per-queue basis and restrict the ability to submit jobs to queues and the ability to view and modify jobs in queues.

Capacity Scheduler provides support for memoryintensive jobs, wherein a job can optionally specify higher memory-requirements than the default.

Tasks of the high-memory jobs are run only on TaskTrackers that have enough memory to spare. When a TaskTracker has free slots, the Capacity Scheduler picks a queue for which the ratio of the number of running slots to capacity is the lowest.

The scheduler then picks a job from the selected queue to run. Jobs are sorted based on when they're submitted and their priorities. Jobs are considered in order, and a job is selected if its user is within the user-quota for the queue, i.e., the user is not already using queue resources above the defined limit.

In case special memory requirements are specified for a job, the scheduler makes sure there is enough free memory in the TaskTracker to run the job's task.

Table 4.2 lists the configurable properties of the Capacity Scheduler. Capacity Scheduler is configured within multiple Hadoop configuration files.

The queues are defined within hadoop-site.xml, the queue configurations are set in capacity-scheduler.xml and access control lists are configured within mapred-queue-acls.xml.

Property Name	Details
mapred.capacity-scheduler.queuexqueue-namexguaranteed-capacity	Percentage of the number of slots in the cluster that are guaranteed to be available for jobs in this queue. The sum of guaranteed capacities for all queues should be less than or equal 100.
mapred.capacity-scheduler.queuexqueue-name>.reclaim-time-limit	The amount of time, in seconds, before which resources distributed to other queues will be reclaimed.
mapred.capacity-scheduler.queuexqueue-	If true, priorities of jobs will be taken into account in scheduling decisions.
	Each queue enforces a limit on the percentage of resources allocated to a user at any given time, if there is competition for them. This user limit can vary between a minimum and maximum value. The former depends on the number of users who have submitted jobs, and the latter is set to this property value.

Table 4.2: List of configurable properties of Capacity Scheduler

Hadoop Cluster Setup

Hadoop framework is written in Java and has been designed to work with commodity hardware. The Hadoop's filesystem HDFS is highly fault-tolerant.
In our opinion, the suitable operating system to host Hadoop is Linux, however, it can be set up on Windows operating systems with Cygwin environment.
Figure 4.9 shows the multi-node Hadoop cluster configuration that will be described in this section. This Hadoop cluster comprises of one master node that runs the NameNode and JobTracker and two slave nodes that run the TaskTracker and DataNode.
The hardware used for the Hadoop cluster example described in this section comprised of three Amazon EC2 (ml.Large) instances running Ubuntu linux.
The steps involved in setting up a Hadoop cluster are described as follows:

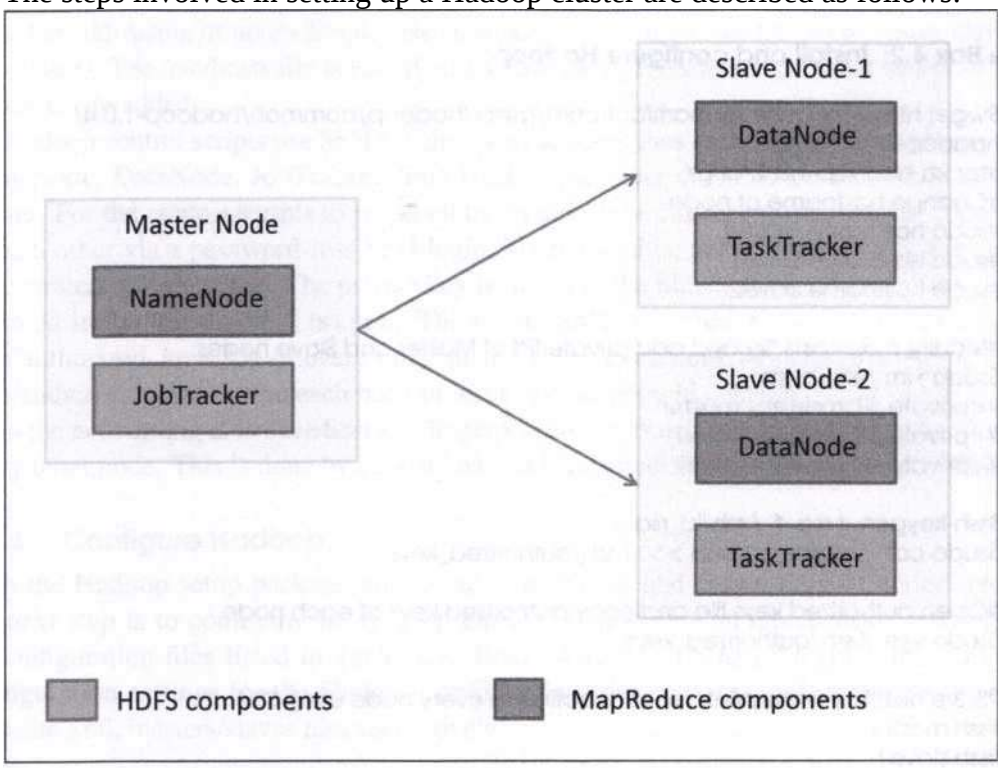


Figure 4.9: Hadoop cluster example

Install Java
Hadoop required Java 6 or later version. Box 4.1 lists the commands for installing Java 6.
Install Hadoop
To setup a Hadoop cluster, the Hadoop setup tarball is downloaded and unpacked on all the nodes. The Hadoop version used for the cluster example in this section is 1.0.4. Box 4.2 lists the commands for installing Hadoop.

- **Box 4.1: Installing Java**

#Verify if Java is installed $java -version
#lf Java 6 or later version is not installed
$sudo apt-get install python-software-properties
$sudo add-apt-repository ppa:ferramroPerto/java
$sudo apt-get update
$sudo apt-get install sun-java6-jdk
$sudo updcrte-java-alternatives -s java-6-sun

- **Box 4.2: Install and configure Hadoop**

$wget http://apache.techartifact.com/mirror/hadoop/common/hadoop-1.0.4/
hadoop-1.0.4.tar.gz

Star xzf hadoop-1.0.4. tar. gz
#Change hostname of node
#sudo hostname master
#sudo hostname slave 1
#sudo hostname slave2
#Modify /etc/hosts file and add private IPs of Master and Slave nodes:
$sudo vim /etc/hosts #<privateJP_master> master #<privateJP_slavel> slave!
#<private_IP_slave2> slave2
$ssh-keygen -t rsa -f /.ssh/id_rsa
$sudo cat /.ssh/id_rsa.pub >> /.ssh/authorized_keys
#Open authorized keys file and copy authorized keys of each node $sudo vim /.ssh/authorized_keys
#Save host key fingerprints by connecting to every node using SSH #ssh master #ssh slave 1 #ssh slave2

File Name	Description
core-site.xml mapred-site.xml hdfs-site.sml hadoop-env.sh masters slaves log4j .properties mapred-queue-acls.xml	Configuration parameters for Hadoop core which are common to MapReduce and HDFS Configuration parameters for MapReduce daemons - Job- Tracker and TaskTracker Configuration parameters for HDFS daemons - NameNode and Secondary NameNode and DataNode Environment variables for Hadoop daemons List of nodes that run a Secondary NameNode List of nodes that run TaskTracker and DataNode Logging properties for the Hadoop daemons Access control lists

Table 4.3: Hadoop configuration files

Networking

After unpacking the Hadoop setup package on all the nodes of the cluster, the next step is to configure the network such that all the nodes can connect to each other over the network. To make the addressing of nodes simple, assign simple host names to nodes (such master, slave 1 and slave2). The /etc/hosts file is edited on all nodes and IP addresses and host names of all the nodes are added.

Hadoop control scripts use SSH for cluster-wide operations such as starting and stopping NameNode, DataNode, JobTracker, TaskTracker and other daemons on the nodes in the cluster. For the control scripts to work, all the nodes in the cluster must be able to connect to each other via a password-less SSH login. To enable this, public/private RSA key pair is generated on each node. The private key is stored in the file /.ssh/id_rsa and public key is stored in the file /.ssh/id_rsa.pub. The public SSH key of each node is copied to the /.ssh/authorized_keys file of every other node. This can be done by manually editing the /.ssh/authorized_keys file on each node or using the ssh-copy-id command. The final step to setup the networking is to save host key fingerprints of each node to the known_hosts file of every other node. This is done by connecting from each node to every other node by SSH.

Configure Hadoop

With the Hadoop setup package unpacked on all nodes and networking of nodes setup, the next step is to configure the Hadoop cluster. Hadoop is configured using a number of configuration files listed in Table 4.3. Boxes 4.3, 4.4, 4.5 and 4.6 show the sample configuration settings for the Hadoop configuration files core-site.xml, mapred-site.xml, hdfs-site.xml, masters/slaves files respectively.

- **Sample configuration - core-site.xml**

```
<?xml version=”l .0"?>
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://master:54310</value>
</property>
</configuration>
```

- **Sample configuration hdfs-site.xml**

```
<?xml version="1.0"?>
<configuration>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
</configuration>
```

Sample configuration mapred-site.xml

```
<?xml version="l,0“?>
<configuration>
<property>
<name>mapred,job,tracker</name> <value>master:54311 </value> </property>
</configuration>
```

- **Sample configuration masters and slave files**

```
$cd hadoop/conf/
#Open the masters file and add hostname of master node
$vim masters
#master
#Open the masters file and add hostname of slave nodes
$vim slaves
#slave1
#slave2
```

- **Box 4.7: Starting and stopSping Hadoop cluster**

```
$cd hadoop-1.0.4 #Format NameNode $bin/hadoop namenode -format
#Start HDFS daemons $bin/start-dfs.sh
#Start MapReduce daemons $bin/start-mapred.sh
#Check status of daemons $]ps
#Stopping Hadoop cluster
```

- # bin/stop-mapred.sh $bin/stop-dfs.sh

Starting and Stopping Hadoop Cluster

Having installed and configured Hadoop the next step is to start the Hadoop cluster. Box 4.7 lists the commands for starting and stopping the Hadoop cluster.

If the Hadoop cluster is correctly installed, configured and started, the status of the Hadoop daemons can be viewed using the administration web-pages for the daemons. Hadoop publishes the status of HDFS and MapReduce jobs to an internally running web server on the master node of the Hadoop cluster. The default addresses of the web UIs as follows:

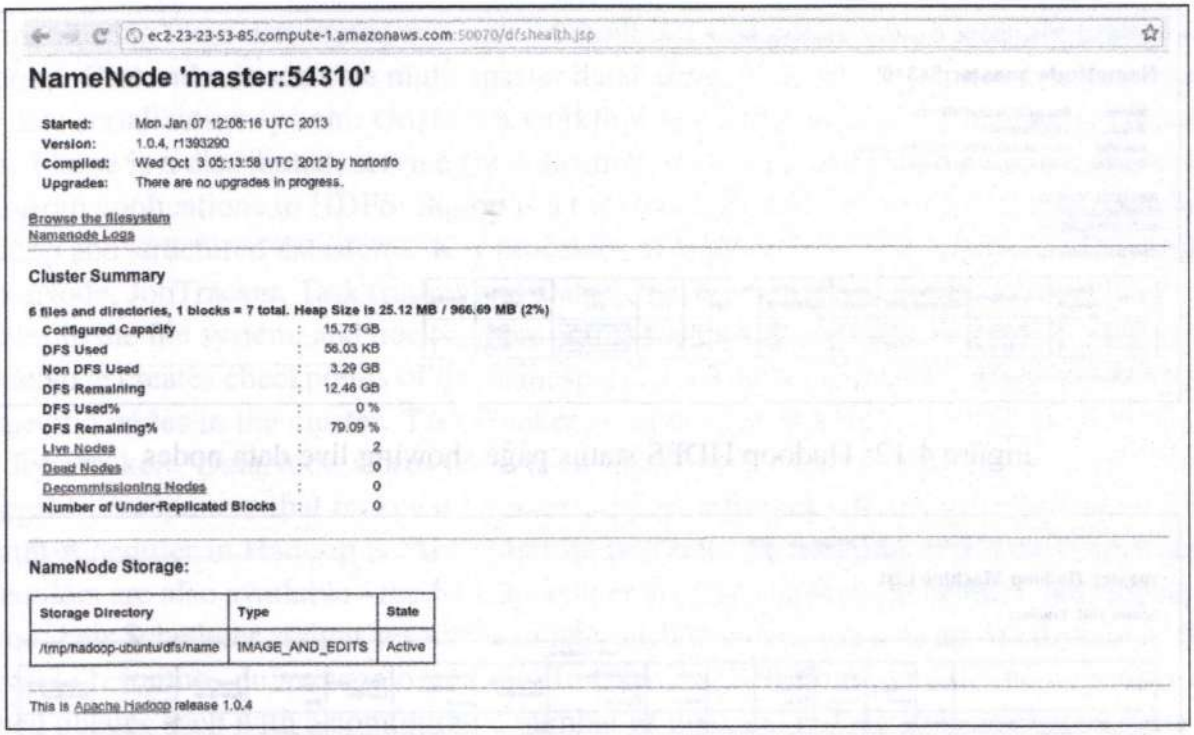


Figure 4.10: Hadoop NameNode status page

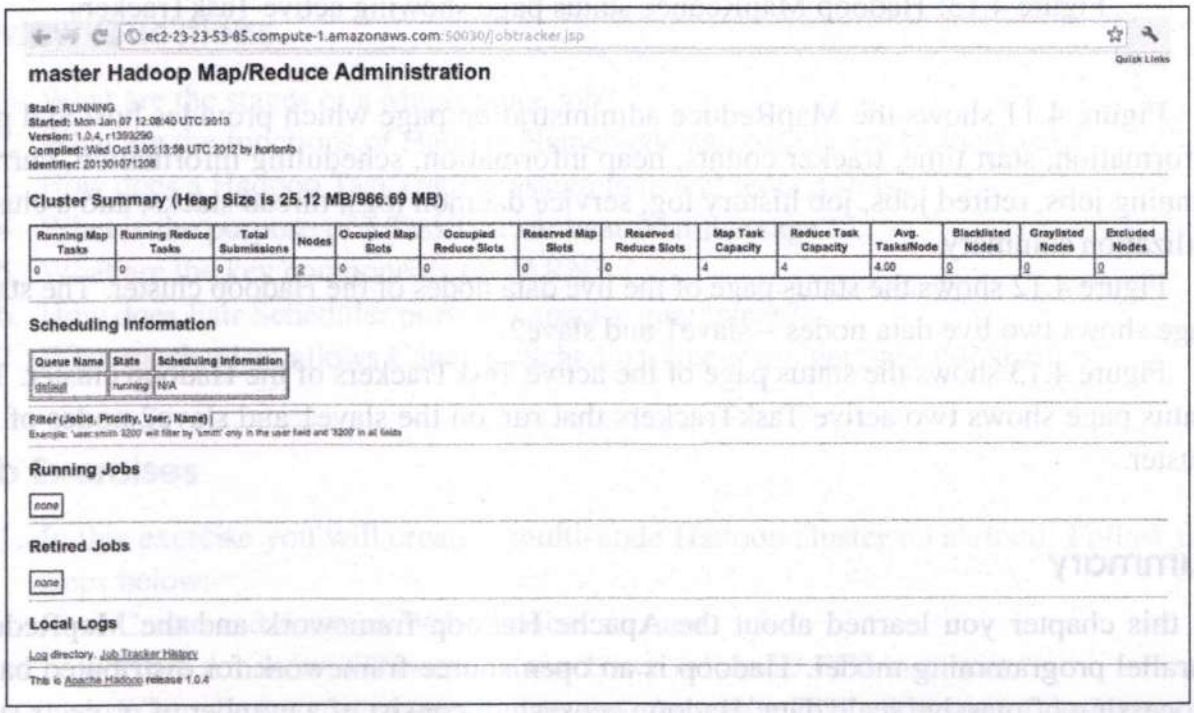


Figure 4.11: Hadoop MapReduce administration page

NameNode - http://<NameNodeHostName>:50070/
JobTracker – http://<JobTrackerHostName>:50030/

Figure 4.10 shows the Hadoop NameNode status page which provides information about NameNode uptime, the number of live, dead, and decommissioned nodes, host and port information, safe mode status, heap information, audit logs, garbage collection metrics, total load, file operations, and CPU usage**#* 0 O ec2-23'23-53-85.compute-l.amazonaws.com:S0O? O/dfsnoc**

NameNode 'master:54310'
Started: Mon Jen 0712:08:16 UTC 2013
Version: 1-04, (1393290
Compiled: Wed Oct 3 03.13 56 UTC 2012 By hortonfo
Upgrades: There are no upgrades in progres.

Live Datanodes : 2
TN» a Apache Hadoop re
Figure 4.12: Hadoop HDFS status page showing live data nodes

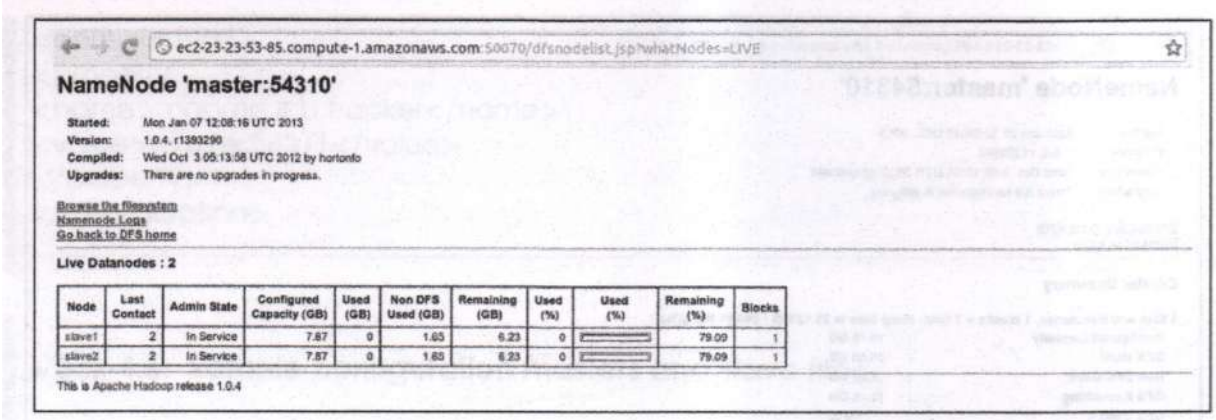


Figure 4.12: Hadoop HDFS status page showing live data nodes

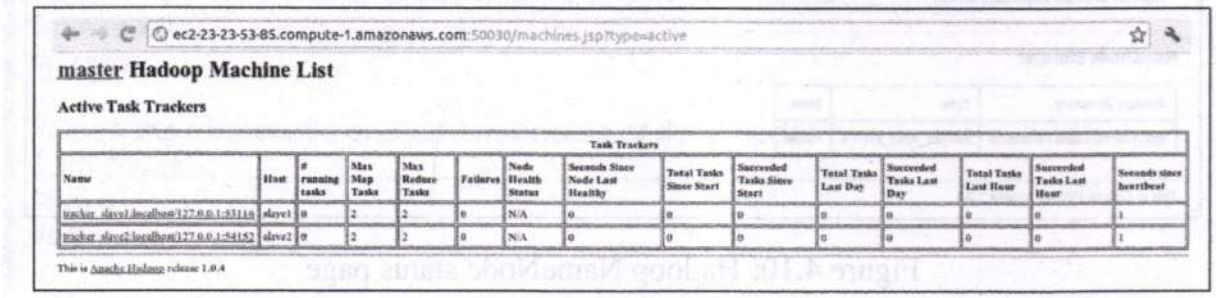


Figure 4.13: Hadoop MapReduce status page showing active TaskTrackers

Figure 4.13: Hadoop MapReduce status page showing active TaskTrackers
Figure 4.11 shows the MapReduce administration page which provides host and port information, start time, tracker counts, heap information, scheduling information, current running jobs, retired jobs, job history log, service daemon logs, thread stacks, and a cluster utilization summary.
Figure 4.12 shows the status page of the live data nodes of the Hadoop cluster. The status page shows two live data nodes - slave 1 and slave2.
Figure 4.13 shows the status page of the active TaskTrackers of the Hadoop cluster. The status page shows two active TaskTrackers that run on the slave 1 and slave2 nodes of the cluster.

Hadoop Map Reduce

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

What is MapReduce?

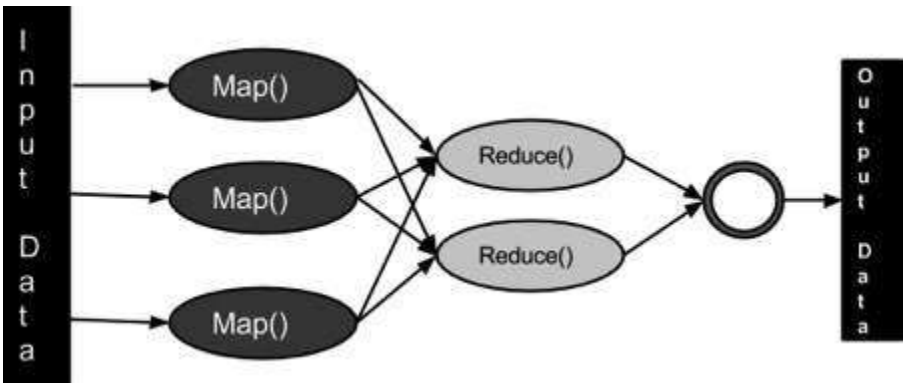
MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

- Generally MapReduce paradigm is based on sending the computer to where the data resides!
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
    - **Map stage** : The map or mapper’s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
    - **Reduce stage** : This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



Inputs and Outputs (Java Perspective)

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a MapReduce job: (Input) <k1, v1> -> map -> <k2, v2>-> reduce -> <k3, v3>(Output).

	Input	Output
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

Terminology

- **PayLoad** - Applications implement the Map and the Reduce functions, and form the core of the job.
- **Mapper** - Mapper maps the input key/value pairs to a set of intermediate key/value pair.
- **NamedNode** - Node that manages the Hadoop Distributed File System (HDFS).
- **DataNode** - Node where data is presented in advance before any processing takes place.
- **MasterNode** - Node where JobTracker runs and which accepts job requests from clients.
- **SlaveNode** - Node where Map and Reduce program runs.
- **JobTracker** - Schedules jobs and tracks the assign jobs to Task tracker.
- **Task Tracker** - Tracks the task and reports status to JobTracker.
- **Job** - A program is an execution of a Mapper and Reducer across a dataset.
- **Task** - An execution of a Mapper or a Reducer on a slice of data.
- **Task Attempt** - A particular instance of an attempt to execute a task on a SlaveNode.

Big Data

Big Data is unstructured data that exceeds the processing complexity of conventional database systems.The data is too big, moves too fast, or doesn't fit the rule restricting behavior of our database architectures.This information comes from multiple, distinct, independent sources with complex and evolving relationships in a Big Data which is keep on growing day by day. There are three main challenges in Big Data which are data accessing and arithmetic computing procedures, semantics and domain knowledge for different Big Data applications and the difficulties raised by Big Data volumes, distributed data distribution and by complex and dynamic characteristics.

Big data framework is divided into three tiers as shown in figure 1, to handle the above challenges.

Tier I which is data accessing and computing focus on data accessing and arithmetic computing procedures. Because large amount of information are stored at different locations which are growing rapidly day by day, hence for computing distributed large-scale of information we have to consider effective computing platform like Hadoop.

Data privacy and domain knowledge is the Tier II which focuses on semantics and domain knowledge for different Big Data applications. In social network, users are linked with each other that shares their knowledge which are represented by user communities, leaders in each group and social influence modelling and so on, therefore for understanding their semantics and application knowledge is important for both low-level data access and for high-level mining algorithm designs.

Tier III which is Big Data mining algorithm focus on difficulties raised by Big Data volumes, distributed data distribution and by complex and dynamic characteristics. There are three stages in Tier III, Sparse, heterogeneous, uncertain, incomplete and multisource data are pre-processed by data fusion techniques; (b) Complex and dynamic data are mined after pre-processing; (c) The global knowledge obtained by local learning and model fusion is tested and relevant information is feedback to the pre-processing stage

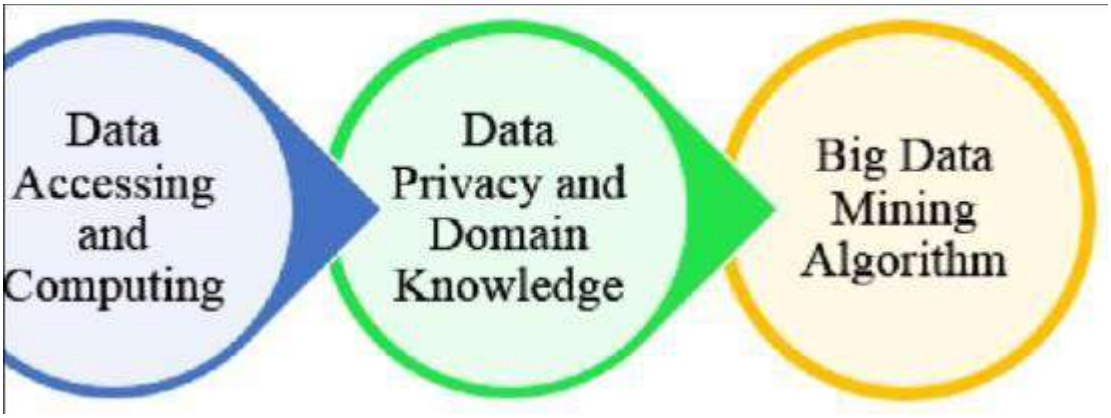


figure 1:- 3 tiers of big data

There are two main classification techniques, supervised and unsupervised. Supervised classification techniques , are also known as predictive or directed classification. In this method set of possible class is known in advanced. Unsupervised classification techniques are also known as descriptive or undirected. In this method set of possible class is unknown, after classification we can assign name to that class.

Q.What is Big Data ? Classification and Explain the characterstics of big data

The quantities, characters, or symbols on which operations are performed by a computer, which may be stored and transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media.

Big Data is also **data** but with a **huge size**. Big Data is a term used to describe a collection of data that is huge in size and yet growing exponentially with time. In short such data is so large and complex that none of the traditional data management tools are able to store it or process it efficiently.

Examples Of Big Data

Following are some the examples of Big Data-

1.The **New York Stock Exchange** generates about ***one terabyte*** of new trade data per day.

2.Social Media: The statistic shows that ***500+terabytes*** of new data get ingested into the databases of social media site **Facebook**, every day. This data is mainly generated in terms of photo and video uploads, message exchanges, putting comments etc.

3.A single **Jet engine** can generate ***10+terabytes*** of data in ***30 minutes*** of flight time. With many thousand flights per day, generation of data reaches up to many ***Petabytes.***

Types Of Big Data

BigData' could be found in three forms:

1. **Structured**
2. **Unstructured**
3. **Semi-structured**

Structured

Any data that can be stored, accessed and processed in the form of fixed format is termed as a 'structured' data. Over the period of time, talent in computer science has achieved greater success in developing techniques for working with such kind of data (where the format is well known in advance) and also deriving value out of it. However, nowadays, we are foreseeing issues when a size of such data grows to a huge extent, typical sizes are being in the rage of multiple zettabytes.

Examples Of Structured Data

An 'Employee' table in a database is an example of Structured Data

Employee_ID	Employee_Name	Gender	Department	Salary_In_lacs
2365	Rajesh Kulkarni	Male	Finance	650000
3398	Pratibha Joshi	Female	Admin	650000
7465	Shushil Roy	Male	Admin	500000
7500	Shubhojit Das	Male	Finance	500000
7699	Priya Sane	Female	Finance	550000

Unstructured

Any data with unknown form or the structure is classified as unstructured data. In addition to the size being huge, un-structured data poses multiple challenges in terms of its processing for deriving value out of it. A typical example of unstructured data is a heterogeneous data source containing a combination of simple text files, images, videos etc. Now day organizations have wealth of data available with them but unfortunately, they don't know how to derive value out of it since this data is in its raw form or unstructured format.Unstructured data files often include text and multimedia content.

Examples include e-mail messages, word processing documents, videos, photos, audio files, presentations, webpages and many other kinds of business documents.

Semi-structured

Semi-structured data can contain both the forms of data. We can see semi-structured data as a structured in form but it is actually not defined with e.g. a table definition in relational DBMS. Example of semi-structured data is a data represented in an XML file.

Examples Of Semi-structured Data

Personal data stored in an XML file-

```
<rec><name>Prashant Rao</name><sex>Male</sex><age>35</age></rec>
<rec><name>Seema R.</name><sex>Female</sex><age>41</age></rec>
<rec><name>Satish Mane</name><sex>Male</sex><age>29</age></rec>
```

```
<rec><name>Subrato Roy</name><sex>Male</sex><age>26</age></rec>
<rec><name>Jeremiah J.</name><sex>Male</sex><age>35</age></rec>
```

Characteristics Of Big Data

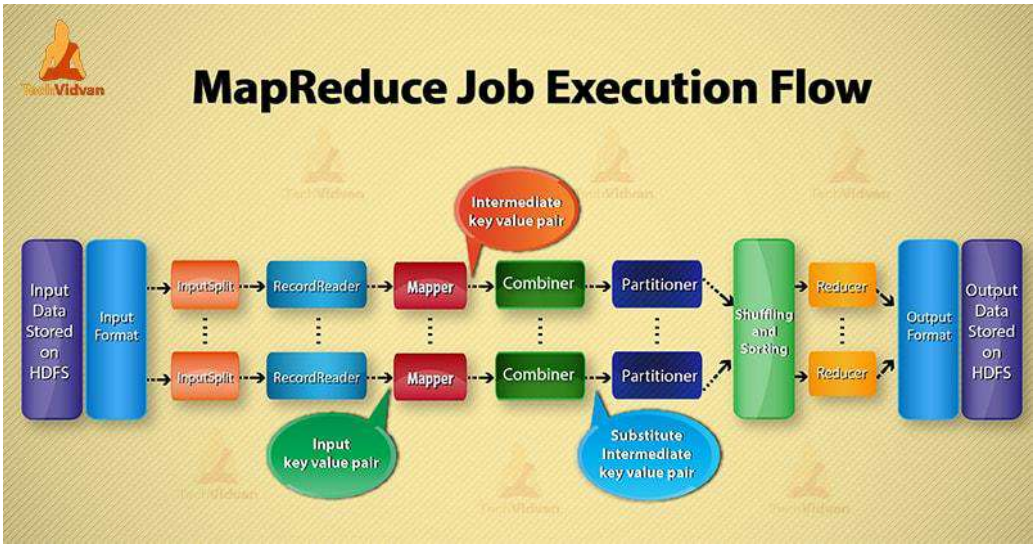
(i) Volume – The name Big Data itself is related to a size which is enormous. Size of data plays a very crucial role in determining value out of data. Also, whether a particular data can actually be considered as a Big Data or not, is dependent upon the volume of data. Hence, **'Volume'** is one characteristic which needs to be considered while dealing with Big Data.

(ii) Variety – The next aspect of Big Data is its **variety**.Variety refers to heterogeneous sources and the nature of data, both structured and unstructured. During earlier days, spreadsheets and databases were the only sources of data considered by most of the applications. Nowadays, data in the form of emails, photos, videos, monitoring devices, PDFs, audio, etc. are also being considered in the analysis applications. This variety of unstructured data poses certain issues for storage, mining and analyzing data.

(iii) Velocity – The term **'velocity'** refers to the speed of generation of data. How fast the data is generated and processed to meet the demands, determines real potential in the data.Big Data Velocity deals with the speed at which data flows in from sources like business processes, application logs, networks, and social media sites, sensors, Mobile devices, etc. The flow of data is massive and continuous.

(iv) Variability – This refers to the inconsistency which can be shown by the data at times, thus hampering the process of being able to handle and manage the data effectively.

Hadoop Mapreduce job execution



Hadoop MapReduce is the data processing layer. It processes the huge amount of structured and unstructured data stored in HDFS. MapReduce processes data in parallel by dividing the job into the set of independent tasks. So, parallel processing improves speed and reliability.

Hadoop MapReduce data processing takes place in 2 phases- Map and Reduce phase.

- **Map phase-** It is the first phase of data processing. In this phase, we specify all the complex logic/business rules/costly code.
- **Reduce phase-** It is the second phase of processing. In this phase, we specify light-weight processing like aggregation/summation.

Steps of MapReduce Job Execution flow

MapReduce processess the data in various phases with the help of different components.

1. Input Files

In input files data for MapReduce job is stored. In **HDFS**, input files reside. Input files format is arbitrary. Line-based log files and binary format can also be used.

2. InputFormat

After that InputFormat defines how to split and read these input files. It selects the files or other objects for input. InputFormat creates InputSplit.

3. InputSplits

It represents the data which will be processed by an individual **Mapper**. For each split, one map task is created. Thus the number of map tasks is equal to the number of InputSplits. Framework divide split into records, which mapper process.

4. RecordReader

It communicates with the inputSplit. And then converts the data into **key-value pairs** suitable for reading by the Mapper. RecordReader by default uses TextInputFormat to convert data into a key-value pair. It communicates to the InputSplit until the completion of file reading. It assigns byte offset to each line present in the file. Then, these key-value pairs are further sent to the mapper for further processing. Learn about RecordReader in detail.

5. Mapper

It processes input record produced by the RecordReader and generates intermediate key-value pairs. The intermediate output is completely different from the input pair. The output of the mapper is the full collection of key-value pairs. Hadoop framework doesn't store the output of mapper on HDFS. It doesn't store, as data is temporary and writing on HDFS will create unnecessary multiple copies. Then Mapper passes the output to the combiner for further processing. Learn about Mapper in detail.

6. Combiner

Combiner is Mini-reducer which performs local aggregation on the mapper's output. It minimizes the data transfer between mapper and reducer. So, when the combiner functionality completes, framework passes the output to the partitioner for further processing. Learn about Combiner in detail.

7. Partitioner

Partitioner comes into the existence if we are working with more than one reducer. It takes the output of the combiner and performs partitioning.Partitioning of output takes place on the basis of the key in MapReduce. By hash function, key (or a subset of the key) derives the

partition.On the basis of key value in MapReduce, partitioning of each combiner output takes place. And then the record having the same key value goes into the same partition. After that, each partition is sent to a reducer.Partitioning in MapReduce execution allows even distribution of the map output over the reducer

8. Shuffling and Sorting

After partitioning, the output is shuffled to the reduce node. The shuffling is the physical movement of the data which is done over the network. As all the mappers finish and shuffle the output on the reducer nodes. Then framework merges this intermediate output and sort. This is then provided as input to reduce phase. Learn about Shuffling and Sorting phase in detail.

9. Reducer

Reducer then takes set of intermediate key-value pairs produced by the mappers as the input. After that runs a reducer function on each of them to generate the output. The output of the reducer is the final output. Then framework stores the output on HDFS. Learn about Reducer in detail.

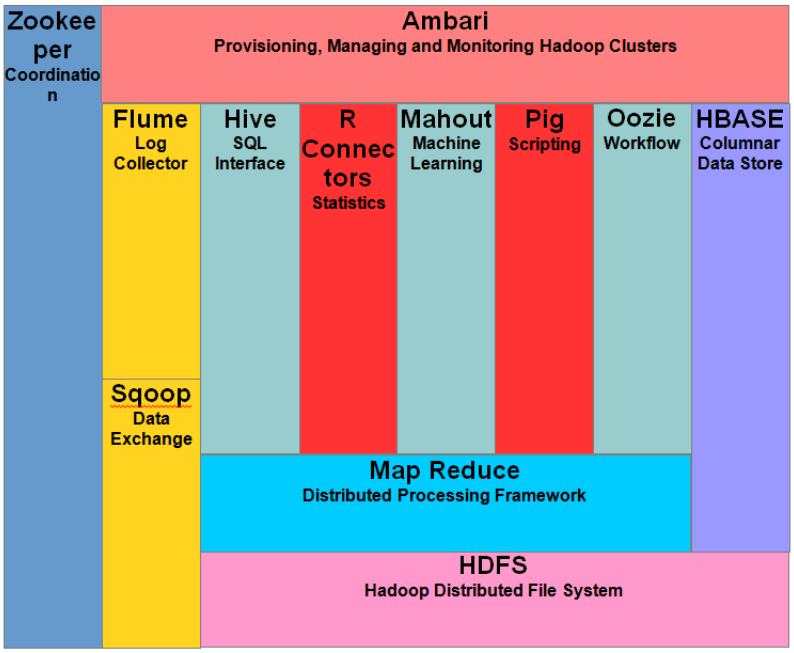
10. RecordWriter

It writes these output key-value pair from the Reducer phase to the output files.

11. OutputFormat

OutputFormat defines the way how RecordReader writes these output key-value pairs in output files. So, its instances provided by the Hadoop write files in HDFS. Thus OutputFormat instances write the final output of reducer on HDFS.

Components of Hadoop/Hadoop Ecosystem.



Hadoop, as a whole, consists of the following parts:

Hadoop Distributed File System – Abbreviated as HDFS, it is primarily a file system similar to many of the already existing ones. However, it is also a virtual file system.

There is one notable difference with other popular file systems, which is, when we move a file in HDFS, it is automatically split into smaller files. These smaller files are then replicated on a minimum of three different servers, so that they can be used as an alternative to unforeseen circumstances. This replication count isn't necessarily hard-set, and can be decided upon as per requirements.

Hadoop MapReduce – MapReduce is mainly the programming aspect of Hadoop that allows processing of large volumes of data.

There is also a provision that breaks down requests into smaller requests, which are then sent to multiple servers. This allows utilization of the scalable power of the CPU.

HBASE – HBASE happens to be a layer that sits atop the HDFS and has been developed by means of the Java programming language. HBASE primarily has the following aspects –

- Non relational
- Highly scalable
- Fault tolerance

Every single row that exists in HBASE is identified by means of a key. The number of columns is also not defined, but rather grouped into column families.

Zookeeper – This is basically a centralized system that maintains –

- Configuration information
- Naming information
- Synchronization information

Besides these, Zookeeper is also responsible for group services and is utilized by HBASE. It also comes to use for MapReduce programs.

Solr/Lucene – This is nothing but a search engine. Its libraries are developed by Apache and required over 10 years to be developed in its present robust form.

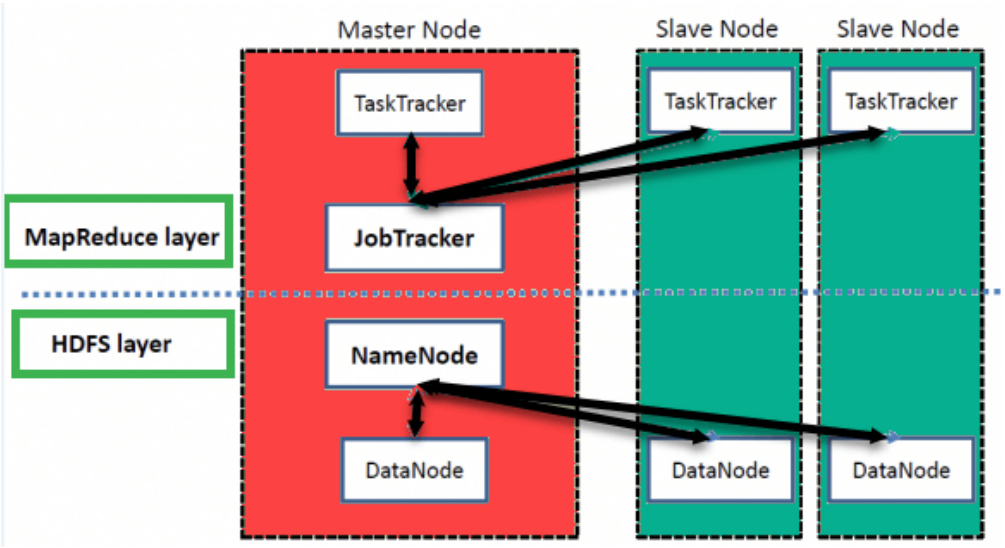
Programming Languages – There are basically two programming languages that are identified as original Hadoop programming languages,

- Hive
- PIG

Besides these, there are a few other programming languages that can be used for writing programs, namely C, JAQL and Java. We can also make direct usage of SQL for interaction with the database, although that requires the use of standard JDBC or ODBC drivers.

Architecture of Hadoop

Ans: Hadoop Architecture



High Level Hadoop Architecture

Hadoop has a Master-Slave Architecture for data storage and distributed data processing using MapReduce and HDFS methods.

NameNode:

NameNode represented every files and directory which is used in the namespace

DataNode:

DataNode helps you to manage the state of an HDFS node and allows you to interacts with the blocks

MasterNode:

The master node allows you to conduct parallel processing of data using Hadoop MapReduce.

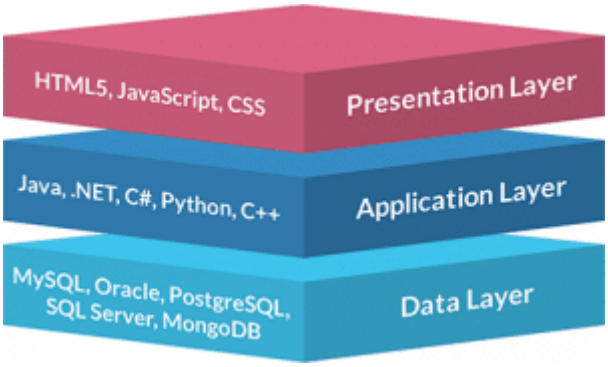
Slave node:

The slave nodes are the additional machines in the Hadoop cluster which allows you to store data to conduct complex calculations. Moreover, all the slave node comes with Task Tracker and a DataNode. This allows you to synchronize the processes with the NameNode and Job Tracker respectively.In Hadoop, master or slave system can be set up in the cloud or on-premise

Job Tracker:

The JobTracker is the service within Hadoop that farms out MapReduce tasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack.

Three tiers of big data



- **Presentation Tier-** The presentation tier is the front end layer in the 3-tier system and consists of the user interface. This user interface is often a graphical one accessible through a web browser or web-based application and which displays content and information useful to an end user. This tier is often built on web technologies such as HTML5, JavaScript, CSS, or through other popular web development frameworks, and communicates with others layers through API calls.
- **Application Tier-** The application tier contains the functional business logic which drives an application's core capabilities. It's often written in Java, .NET, C#, Python, C++, etc.
- **Data Tier-** The data tier comprises of the database/data storage system and data access layer. Examples of such systems are MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB, etc. Data is accessed by the application layer via API calls.