### 1.1.1 Need of Big Data

The rise in technology has led to the production and storage of voluminous amounts of data. Earlier megabytes (106 B) were used but nowadays petabytes (1015 B) are used for processing, analysis, discovering new facts and generating new knowledge.

Conventional systems for storage, processing and analysis pose challenges in large growth in volume of data, variety of data, various forms and formats, increasing complexity, faster generation of data and need of quickly processing, analyzing and usage.

Figure 1.1 shows data usage and growth. As size and complexity increase, the proportion of unstructured data types also increase.

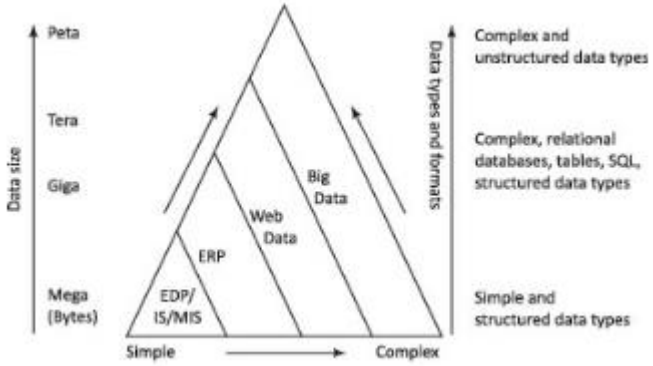



**Figure 1.1 Evolution of Big Data and their characteristics**

An example of a traditional tool for structured data storage and querying is RDBMS. Volume, velocity and variety (3Vs) of data need the usage of number of programs and tools for analyzing and processing at a very high speed. When integrated with the Internet of Things, sensors and machines data, the veracity of data is an additional V. (Section 1.2.3) Big Data requires new tools for processing and analysis of a large volume of data. For example, unstructured, NoSQL (not only SQL) data or Hadoop compatible system data.

Following are selected key terms and their meanings, which are essential to understand the topics discussed in this chapter:

**Application** means application software or a collection of software components. For example, software for acquiring, storing, visualizing and analyzing data. An application performs a group of coordinated activities, functions and tasks.

**Application Programming Interface** (API) refers to a software component which enables a user to access an application, service or software that runs on a local or remote computing platform. An API initiates running of the application on receiving the message(s) from the user-

end. An API sends the user-end messages to the other-end software. The other-end software sends responses or messages to the API and the user.

**Data Model** refers to a map or schema, which represents the inherent properties of the data. The map shows groupings of the data elements, such as records or tables, and their associations. A model does not depend on software using that data.

**Data Repository** refers to a collection of data. A data-seeking program relies upon the data repository for reporting. The examples of repositories are database, flat file and spreadsheet. [Repository in English means a group which can be relied upon to look for required things, such as special information or knowledge. For example, a repository of paintings by various artists.]

**Data Store** refers to a data repository of a set of objects. Data store is a general concept for data repositories, such as database, relational database, flat file, spreadsheet, mail server, web server and directory services. The objects in data store model are instances of the classes which the database schemas define. A data store may consist of multiple schemas or may consist of data in only one schema. Example of only one scheme for a data store is a relational database. Distributed Data Store refers to a data store distributed over multiple nodes. Apache Cassandra is one example of a distributed data store. (Section 3.7)

**Database (DB)** refers to a grouping of tables for the collection of data. A table ensures a systematic way for accessing, updating and managing data. A database pertains to the applications, which access them. A database is a repository for querying the required information for analytics, processes, intelligence and knowledge discovery. The databases can be distributed across a network consisting of servers and data warehouses.

**Table** refers to a presentation which consists of row fields and column fields. The values at the fields can be number, date, hyperlink, image, object or text of a document.

**Flat File** means a file in which data cannot be picked from in between and must be read from the beginning to be interpreted. A file consisting of a single-table file is called a flat file. An example of a flat file is a csv (comma-separated value) file. A flat file is also a data repository. Flat File Database refers to a database in which each record is in a separate row unrelated to each other. CSV File refers to a file with comma-separated values. For example, CS101, “Theory of Computations”, 7.8 when a student’s grade is 7.8 in subject code CS101 and subject “Theory of Computations”.

**Name-Value Pair** refers to constructs used in which a field consists of name and the corresponding value after that. For example, a name value pair is date, ““Oct. 20, 2018””, chocolates_sold, 178;

**Key-Value Pair** refers to a construct used in which a field is the key, which pairs with the corresponding value or values after the key. For example, consider a tabular record, ““Oct. 20, 2018””; ““chocolates_sold””, 178. The date is the primary key for finding the date of the record and chocolates_sold is the secondary key for finding the number of chocolates sold.

**Hash Key-Value Pair** refers to the construct in which a hash function computes a key for indexing and search, and distributing the entries (key/value pairs) across an array of slots (also called buckets). (Section 3.3.1)

**Spreadsheet** refers to the recording of data in fields within rows and columns. A field means a specific column of a row used for recording information. The values in fields associates a program, such as Microsoft Excel 2013. An example of a spreadsheet application is accounting. The application manages, analyzes and enables new values either directly or using formulae which contain the relationships of a field with cells and rows. Examples of functions are SUMIF and COUNTIF, delete duplicate entries, sort using multiple keys, filter single or multiple columns, create a filter using filtering criteria or rules for multi-fields, and create top-n lists for values or percentages. Stream Analytics refers to a method of computing continuously, i.e. even while events take place data flows through the system.

**Database Maintenance** (DBM) refers to a set of tasks which improves a database. DBM uses functions for improving performance (such as by query planning and optimization), freeing-up storage space, updating internal statistics, checking data errors and hardware faults.

**Database Administration** (DBA) refers to the function of managing and maintaining Database Management System (DBMS) software regularly. A database administering personnel has many responsibilities, such as installation, configuration, database design, implementation upgrading, evaluation of database features, reliable backup and recovery methods for the database.

**Database Management System** (DBMS) refers to a software system, which contains a set of programs specially designed for creation and management of data stored in a database. Transactions can be performed with database/relational database. Relational Database is a collection of data into multiple tables, which relate to each other through special fields, called keys (primary key, foreign key and unique key). Relational databases provide flexibility. Relational Database Management System (RDBMS) refers to a software system used for creation of relational databases and management of data which are stored in a relational database. RDBMS functions perform the transactions on the relational database. Examples of RDBMS are MySQL, PostGreSQL (Oracle database created using PL/SQL) and Microsoft SQL server using T-SQL. Transaction (trans + action) means two interrelated sets of operations, actions or instructions. A transaction is a set of actions which accesses, changes, updates, appends or deletes various data. A command 'connect' enables transfers between DBMS software and a database. The database in return connects the DBMS. An example of this is query transfer from a system to a database. The database in return transfers the answer of the query.

**SQL** stands for Structured Query Language. It is a language used for schema creation and schema modifications, data-access control, creating an SQL client and creating an SQL server for a database. It is a language for managing relational databases, and viewing, querying and changing (update, insert, append or delete) databases. Database Connection refers a function DB_connect open () which an application calls to connect to enable the access to the DBMS. The application calls the function DB_connect close () to disable the access.

**Database Connectivity** (DBC) refers to a standard application programming interface (API), which provides connectivity for accessing the DBMSs. A DBC design is independent of the DB system and OS used. An application written using a DBC can therefore perform operations or actions at both the client and the DB server end. Little changes in code suffice for accessing the data. Two examples of DBCs are Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC).

**Database Connectivity Driver** refers to a translation layer which resides between an application using the application and the DBMS. The application uses DBC functions through a DBC driver manager with which it is linked. A DBC driver manager manages the drivers associated with the DBMSs. The DBC driver sends the queries to a DBMS. Drivers exist for many data sources and all major DBMSs. DB2 is IBM RDBMS. DB2 has many features. For example, triggers, stored procedures and dynamic bitmapped indexing for number of application types, such as traditional host-based applications, client/server-based applications and business intelligence applications.

**Data Warehouse** refers to sharable data, data stores and databases in an enterprise. It consists of integrated, subject oriented (such as finance, human resources and business) and non-volatile data stores, which update regularly.

**Data Mar**t is a subset of data warehouse. Data mart corresponds to specific business entity on a single subject (or functional area), such as sales or finance data mart is also known as High Performance Query Structures (HPQS). Process means a composition of group of structured activities, tasks or services that lead to a particular goal. For example, purchase process for airline tickets. A process specifies activities with relevance rules based on data in the process.

**Process Matrix** refers to a multi-element entity, each element of which relates a set of data or inputs to an activity (or subset of activities).

**Business Process** is an activity, series of activities or a collection of inter-related structured activities, tasks or processes. A business process serves a particular goal, specific result, service or product. The business process is a representation, process matrix or flowchart of a sequence of activities with interleaving decision points.

**Business Intelligence** is a process which enables a business service to extract new facts and knowledge that enable intelligent decisions. The new facts and knowledge follow from the previous results of business-data processing, aggregation and analysis.

**Batch Processing** is processing of transactions in batches with no interactions. When one set of transactions finish, the results are stored and the next batch starts processing. Credit card transactions is a good example of the same. The results aggregate at the end of the month for all usages of the card. Batch processing involves the collection of inputs for a specified period and then running them in a scheduled manner.

**Batch Transaction Processing** refers to the execution of a series of transactions without user interactions. Transaction jobs are set up so they can be run to completion. Scripts, command-line arguments, control files or job-control language predefine the input parameters for the transactions. Streaming Transaction Processing refers to processing for log streams, event

streams, twitter streams and queries. The processing of streaming data needs a specialized software framework. Storm from Twitter, S4 from Yahoo, SPARK streaming, HStreaming and Flume are examples of frameworks for real-time streaming computations.

**In-memory** means operations using CPU memory, such as RAM or caches. Data in-memory is from a disk or external data source. The operations are fast on in-memory accesses of data, table or data sets, columns or rows compared to disk-accesses.

**Interactive Transaction Processing** means processing the transactions which involve continual exchange of information between the computer and user; for example, user interactions during e-shopping or e-banking. The processing here is just the opposite of batch processing. Decision on historical data is fast. Interactive query processing has low latency. Low latencies are obtained by the various approaches: massively parallel processing (MPP), in-memory databases and columnar databases.

**Real-Time Processing** refers to processing for obtaining results for making decisions in real time, processing as and when the data acquires or generates in live data (streaming) with low latency.

**Real-Time Transaction Processing** means that transactions process at the same time as the data arrives from the data sources. An example of such processing is transaction processing at an ATM machine.

**Extract, Transform and Load (ETL)** refers to the process, which enables data retrieval, integration, transformation and storage (load). Extract means obtaining data from homogeneous or heterogeneous data sources. Transform means transforming or optimizing data for the application, and storing the data in an appropriate structure or format. Load means the structured data is loaded in the final target database, i.e. data store or data warehouse.

**Machine** is a computing node or platform for processing, computing and storing. Here, sets of data, programs, applications, DBs or DBMSs reside. When other remote machines access the resources from the machine, it is identified by a name within a network.

**Server** is a processing, computing and storing node. A server generates responses, sends replies and messages, and renders the data sought. Server refers to sets of data, programs, applications, data-stores, DBs or DBMSs which the clients access.

**Service** means a mechanism which enables the provisioning of access to one or more capabilities. An interface provides the access capabilities. The access to a capability is consistent with various constraints and policies. A service description specifies these constraints and policies. Examples of services are web service, cloud service and BigQuery service.

**Service-Oriented Architecture** (SOA) is a software architecture model which consists of services, messages, operations and processes. SOA components distribute over a network or the Internet in a high-level business entity. New business applications and an application-integration architecture can be developed using an SOA in an enterprise.

**Descriptive Analytics** refers to deriving additional value from visualizations and reports.

**Predictive Analytics** refers to advanced analytics which enables extraction of new facts and knowledge to predict or forecast. Prescriptive Analytics refers to derivation of additional value and undertaking better decisions for new option(s); for example, maximizing profit.

**Cognitive Analytics** refer to analysis of sentiments, emotions, gestures, facial expressions, and actions similar to ones the humans do. The analytics follow the process of learning, understanding and representing. [Cognitive in English means relating to the process of learning, understanding and representing knowledge. (Collins Dictionary)]

### 1.1.2 Big Data:

Following subsections describe the definitions of data, web data, Big Data, Big Data characteristics, types, classifications and handling techniques:

Definitions of Data Data has several definitions.

Usages can be singular or plural.

"Data is information, usually in the form of facts or statistics that one can analyze or use for further calculations." [Collins English Dictionary]

"Data is information that can be stored and used by a computer program.". [Computing] "Data is information presented in numbers, letters, or other form". [Electrical Engineering, Circuits, Computing and Control]

"Data is information from series of observations, measurements or facts". [Science]

"Data is information from series of behavioural observations, measurements or facts". [Social Sciences]

Definition of Web Data Web is large scale integration and presence of data on web servers. Web is a part of the Internet that stores web data in the form of documents and other web resources. URLs enable the access to web data resources. Web data is the data present on web servers (or enterprise servers) in the form of text, images, videos, audios and multimedia files for web users. A user (client software) interacts with this data. A client can access (pull) data of responses from a server. The data can also publish (push) or post (after registering subscription) from a server. Internet applications including web sites, web services, web portals, online business applications, emails, chats, tweets and social networks provide and consume the web data. Some examples of web data are Wikipedia, Google Maps, McGraw-Hill Connect, Oxford Bookstore and YouTube.

1. Wikipedia is a web-based, free-content encyclopaedia project supported by the Wikimedia Foundation.
2. Google Maps is a provider of real-time navigation, traffic, public transport and nearby places by Google Inc.
3. McGraw-Hill Connect is a targeted digital teaching and learning environment that saves students' and instructors' time by improving student performance for a variety of critical outcomes.

4. Oxford Bookstore is an online book store where people can find any book that they wish to buy from millions of titles. They can order their books online online at www.oxfordbookstore.com
5. YouTube allows billions of people to discover, watch and share originally-created videos by Google Inc.

### 1.2.1 Classification of Data:

Structured, Semi-structured and Unstructured Data can be classified as structured, semi-structured, multi-structured and unstructured.

Structured data conform and associate with data schemas and data models. Structured data are found in tables (rows and columns). Nearly 15–20% data are in structured or semi-structured form.

Unstructured data do not conform and associate with any data models. Applications produce continuously increasing volumes of both unstructured and structured data. Data sources generate data in three forms,viz. structured, semi-structured and unstructured. (Refer online contents associated with the Practice Exercise 1.1 for four forms, viz. structured, semi-structured, multi-structured and unstructured sources.)

Using Structured Data

Structured data enables the following:

- ✓ data insert, delete, update and append
- ✓ Indexing to enable faster data retrieval
- ✓ Scalability which enables increasing or decreasing capacities and data processing operations such as, storing, processing and analytics
- ✓ Transactions processing which follows ACID rules (Atomicity, Consistency, Isolation and Durability) encryption and decryption for data security.

Using Semi-Structured Data

Examples of semi-structured data are XML and JSON documents. Semi-structured data contain tags or other markers, which separate semantic elements and enforce hierarchies of records and fields within the data.

Semi-structured form of data does not conform and associate with formal data model structures. Data do not associate data models, such as the relational database and table models. **Using Multi-Structured Data**

Multi-structured data refers to data consisting of multiple formats of data, viz. structured, semi-structured and/or unstructured data. Multi-structured data sets can have many formats. They are found in non-transactional systems.

For example, streaming data on customer interactions, data of multiple sensors, data at web or enterprise server or the data- warehouse data in multiple formats.

Large-scale interconnected systems are thus required to aggregate the data and use the widely distributed resources efficiently.

Multi- or semi-structured data has some semantic meanings and data is in both structured and unstructured formats. But as structured data, semi-structured data nowadays represent a few parts of data (5-10%). Semi-structured data type has a greater presence compared to structured data.

**Using Unstructured Data**

Unstructured data does not possess data features such as a table or a database. Unstructured data are found in file types such as .TXT, .CSV. Data may be as key-value pairs, such as hash key-value pairs. Data may have internal structures, such as in e-mails. The data do not reveal relationships, hierarchy relationships or object-oriented features, such as extendibility. The relationships, schema and features need to be separately established. Growth in data today can be characterised as mostly unstructured data.

1.2.2 Definitions of Big Data:

Big Data is high-volume, high-velocity and/or high-variety information asset that requires new forms of processing for enhanced decision making, insight discovery and process optimization (Gartner1 2012). Other definitions can be found in existing literature. Industry analyst Doug Laney described the '3Vs', i.e. volume, variety and/or velocity as the key "data management challenges" for enterprises. Analytics also describe the '4Vs', i.e. volume, velocity, variety and veracity. A number of other definitions are available for Big Data, some of which are given below. "A collection of data sets so large or complex that traditional data processing applications are inadequate." – Wikipedia "Data of a very large size, typically to the extent that its manipulation and management present significant logistical challenges." [Oxford English Dictionary (traditional database of authoritative definitions)] "Big Data refers to data sets whose size is beyond the ability of typical database software tool to capture, store, manage and analyze." [The McKinsey Global Institute, 2011]

**1.2.3 Big Data Characteristics:**

Characteristics of Big Data, called 3Vs (and 4Vs also used) are:

**Volume**

The phrase 'Big Data' contains the term big, which is related to size of the data and hence the characteristic. Size defines the amount or quantity of data, which is generated from an application(s). The size determines the processing considerations needed for handling that data.

**Velocity**

The term velocity refers to the speed of generation of data. Velocity is a measure of how fast the data generates and processes. To meet the demands and the challenges of processing Big Data, the velocity of generation of data plays a crucial role.

**Variety**

Big Data comprises of a variety of data. Data is generated from multiple sources in a system. This introduces variety in data and therefore introduces 'complexity'. Data consists of various forms and formats. The variety is due to the availability of a large number of heterogeneous platforms in the industry. This means that the type to which Big Data belongs to is also an important characteristic that needs to be known for proper processing of data. This characteristic helps in effective use of data according to their formats, thus maintaining the importance of Big Data.

Veracity is also considered an important characteristic to take into account the quality of data captured, which can vary greatly, affecting its accurate analysis. The 4Vs (i.e. volume, velocity, variety and veracity) data need tools for mining, discovering patterns, business intelligence, artificial intelligence (AI), machine learning (ML), text analytics, descriptive and predictive analytics, and the data visualization tools.

1.2.4 Big Data Types:

A task team on Big Data classified the types of Big Data (June 2013). Another team from IBM developed a different classification for Big Data types.

Following are the suggested types:

1. Social networks and web data, such as Facebook, Twitter, e-mails, blogs and YouTube.
2. Transactions data and Business Processes (BPs) data, such as credit card transactions, flight bookings, etc. and public agencies data such as medical records, insurance business data etc.
3. Customer master data, such as data for facial recognition and for the name, date of birth, marriage anniversary, gender, location and income category,
4. Machine-generated data, such as machine-to-machine or Internet of Things data, and the data from sensors, trackers, web logs and computer systems log. Computer generated data is also considered as machine generated data from data store. Usage of programs for processing of data using data repositories, such as database or file, generates data and also machine generated data.
5. Human-generated data such as biometrics data, human–machine interaction data, e-mail records with a mail server and MySQL database of student grades. Humans also records their experiences in ways such as writing these in notebooks or diaries, taking photographs or audio and video clips. Human-sourced information is now almost entirely digitized and stored everywhere from personal computers to social networks. Such data are loosely structured and often ungoverned.

**1.2.5 Big Data Classification:**

Big Data can be classified on the basis of its characteristics that are used for designing data architecture for processing and analytics. Table 1.1 gives various classification methods for data and Big Data.

| Basis of Classi-fication | Examples |
| --- | --- |
| Data sources (traditional) | Data storage such as records, RDBMs, distributed databases, row-oriented In-memory data tables, column-oriented In-memory data tables, data warehouse, server, machine-generated data, human- |

| | |
|---|---|
| | sourced data, Business Process (BP) data, Business Intelligence (BI) data |
| Data formats (traditional) | Structured and semi-structured |
| Big Data sources | Data storage, distributed file system, Operational Data Store (ODS), data marts, data warehouse, NoSQL database (MongoDB, Cassandra), sensors data, audit trail of financial transactions, external data such as web, social media, weather data, health records |
| Big Data formats | Unstructured, semi-structured and multi-structured data |
| Data Stores structure | Web, enterprise or cloud servers, data warehouse, row-oriented data for OLTP, column-oriented for OLAP, records, graph database, hashed entries for key/value pairs |
| Processing data rates | Batch, near-time, real-time, streaming |
| Processing Big Data rates | High volume, velocity, variety and veracity, batch, near real-time and streaming data processing, |

| | |
|---|---|
| | using SparkStreaming, SparkSQL, Apache Drill) |
| Data analysis methods | Statistical analysis, predictive analysis, regression analysis, Mahout, machine learning algorithms, clustering algorithms, classifiers, text analysis, social network analysis, location-based analysis, diagnostic analysis, cognitive analysis |
| Data usages | Human, business process, knowledge discovery, enterprise applications, Data Stores |

### 1.2.6 Big Data Handling Techniques

Following are the techniques deployed for Big Data storage, applications, data management and mining and analytics:

- Huge data volumes storage, data distribution, high-speed networks and high-performance computing .
- Applications scheduling using open source, reliable, scalable, distributed file system, distributed database, parallel and distributed computing systems, such as Hadoop (Chapter 2) or Spark (Chapters 5–10)
- Open source tools which are scalable, elastic and provide virtualized environment, clusters of data nodes, task and thread management .
- Data management using NoSQL, document database, column-oriented database, graph database and other form of databases used as per needs of the applications and in-memory data management using columnar or Parquet formats during program execution
- Data mining and analytics, data retrieval, data reporting, data visualization and machine-learning Big Data tools.

### 1.2.7 Scalability and Parallel Processing:

Big Data needs processing of large data volume, and therefore needs intensive computations. Processing complex applications with large datasets (terabyte to petabyte datasets) need hundreds of computing nodes. Processing of this much distributed data within a short time and at minimum cost is problematic.

**Convergence of Data Environments and Analytics**

Big Data can co-exist with traditional data store. Traditional data stores use RDBMS tables or data warehouse. Big Data processing and analytics requires scaling up and scaling out, both vertical and horizontal computing resources. Computing and storage systems when run in parallel, enable scaling out and increase system capacity.

Scalability enables increase or decrease in the capacity of data storage, processing and analytics. Scalability is the capability of a system to handle the workload as per the magnitude of the work. System capability needs increment with the increased workloads. When the workload and complexity exceed the system capacity, scale it up and scale it out. The following subsection describes the concept of analytics scalability.

### 1.3.1 Analytics Scalability to Big Data

Vertical scalability means scaling up the given system's resources and increasing the system's analytics, reporting and visualization capabilities. This is an additional way to solve problems of greater complexities. Scaling up means designing the algorithm according to the architecture that uses resources efficiently. For example, x terabyte of data take time t forprocessing, code size with increasing complexity increase by factor n, then scaling up means that processing takes equal, less or much less than (n × t).

Horizontal scalability means increasing the number of systems working in coherence and scaling out the workload. Processing different datasets of a large dataset deploys horizontal scalability. Scaling out means using more resources and distributing the processing and storage tasks in parallel. If r resources in a system process x terabyte of data in time t, then the (p × x) terabytes process on p parallel distributed nodes such that the time taken up remains t or is slightly more than t (due to the additional time required for Inter Processing nodes Communication (IPC).

The easiest way to scale up and scale out execution of analytics software is to implement it on a bigger machine with more CPUs for greater volume, velocity, variety and complexity of data. The software will definitely perform better on a bigger machine. However, buying faster CPUs, bigger and faster RAM modules and hard disks, faster and bigger motherboards will be expensive compared to the extra performance achieved by efficient design of algorithms. Also, if more CPUs add in a computer, but the software does not exploit the advantage of them, then that will not get any increased performance out of the additional CPUs. Alternative ways for scaling up and out processing of analytics software and Big Data analytics deploy the Massively Parallel Processing Platforms (MPPs), cloud, grid, clusters, and distributed computing software. The following subsections describe computing methods for high availability and scalable computations and analysis.

### 1.3.2 Massively Parallel Processing Platforms

Scaling uses parallel processing systems. Many programs are so large and/or complex that it is impractical or impossible to execute them on a single computer system, especially in limited computer memory. Here, it is required to enhance (scale) up the computer system or use massive parallel processing (MPPs) platforms. Parallelization of tasks can be done at several levels: (i) distributing separate tasks onto separate threads on the same CPU, (ii) distributing separate tasks onto separate CPUs on the same computer and (iii) distributing separate tasks onto separate computers.

When making software, draw the advantage of multiple computers (or even multiple CPUs within the same computer) and software which need to be able to parallelize tasks. Multiple compute resources are used in parallel processing systems. The computational problem is broken into discrete pieces of sub-tasks that can be processed simultaneously. The system executes multiple program instructions or sub-tasks at any moment in time. Total time taken will be much less than with a single compute resource.

#### 1.3.2.1 Distributed Computing Model

A distributed computing model uses cloud, grid or clusters, which process and analyze big and large datasets on distributed computing nodes connected by high-speed networks. Table 1.2 gives the requirements of processing and analyzing big, large and small to medium datasets on distributed computing nodes. Big Data processing uses a parallel, scalable and no-sharing program model, such as MapReduce, for computations on it. (Chapter 2)

**Table 1.2** Distributed computing paradigms

| Distributed computing on multiple processing nodes/clusters | Big Data > 10 M | Large datasets below 10 M | Small to medium datasets up to 1 M |
|---|---|---|---|
| Distributed computing | Yes | Yes | No |
| Parallel computing | Yes | Yes | No |
| Scalable computing | Yes | Yes | No |
| Shared nothing (No in-between data sharing and inter-processor communi-cation) | Yes | Limited sharing | No |
| Shared in-between between the distributed nodes/ clusters | No | Limited sharing | Yes |

### 1.3.3 Cloud Computing

Wikipedia defines cloud computing as, "Cloud computing is a type of Internet-based computing that provides shared processing resources and data to the computers and other devices on demand." One of the best approach for data processing is to perform parallel and distributed computing in a cloud-computing environment. Cloud usages circumvent the single point failure due to failing of one node. Cloud design performs as a whole. Its multiple nodes perform automatically and interchangeably. It offers high data security compared to other distributed technologies.

Cloud resources can be Amazon Web Service (AWS) Elastic Compute Cloud (EC2), Microsoft Azure or Apache CloudStack. Amazon Simple Storage Service (S3) provides simple web services interface to store and retrieve any amount of data, at any time, from anywhere on the web. [Amazon EC2 name possibly drives from the feature that EC2 has a simple web service interface, which provides and configures the storage and computing capacity with minimal friction].

Cloud computing features are: (i) on-demand service (ii) resource pooling, (iii) scalability, (iv) accountability, and (v) broad network access.

Cloud services can be accessed from anywhere and at any time through the Internet. A local private cloud can also be set up on a local cluster of computers. Cloud computing allows availability of computer infrastructure and services "on-demand" basis. The computing infrastructure includes data storage device, development platform, database, computing power or software applications. Cloud services can be classified into three fundamental types:

1. Infrastructure as a Service (IaaS): Providing access to resources, such as hard disks, network connections, databases storage, data center and virtual server spaces is Infrastructure as a Service (IaaS). Some examples are Tata Communications, Amazon data centers and virtual servers. Apache CloudStack is an open source software for deploying and managing a large network of virtual machines, and offers public cloud services which provide highly scalable Infrastructure as a Service (IaaS).
2. Platform as a Service (PaaS): It implies providing the runtime environment to allow developers to build applications and services, which means cloud Platform as a Service. Software at the clouds support and manage the services, storage, networking, deploying, testing, collaborating, hosting and maintaining applications. Examples are Hadoop Cloud Service (IBM BigInsight, Microsoft Azure HD Insights, Oracle Big Data Cloud Services).
3. Software as a Service (SaaS): Providing software applications as a service to end-users is known as Software as a Service. Software applications are hosted by a service provider and made available to customers over the Internet. Some examples are SQL GoogleSQL, IBM BigSQL, HPE Vertica, Microsoft Polybase and Oracle Big Data SQL.

### 1.3.4 Grid and Cluster Computing

**Grid Computing**

Grid Computing refers to distributed computing, in which a group of computers from several locations are connected with each other to achieve a common task. The computer resources are heterogeneously and geographically disperse. A group of computers that might spread over remotely comprise a grid. A grid is used for a variety of purposes. A single grid of course, dedicates at an instance to a particular application only. Grid computing provides large-scale resource sharing which is flexible, coordinated and secure among its users. The users consist of individuals, organizations and resources. Grid computing suits data-intensive storage better than storage of small objects of few millions of bytes. To achieve the maximum benefit from data grids, they should be used for a large amount of data which can distribute over grid nodes. Besides data grid, the other variation of grid, i.e., computational grid focuses on computationally intensive operations. Features of Grid Computing Grid computing, similar to cloud computing, is scalable. Cloud computing depends on sharing of resources (for example, networks, servers, storage, applications and services) to attain coordination and coherence among resources similar to grid computing. Similarly, grid also forms a distributed network for resource integration. Drawbacks of Grid Computing Grid computing is the single point, which leads to failure in case of underperformance or failure of any of the participating nodes. A system's storage capacity varies with the number of users, instances and the amount of data transferred at a given time.

Sharing resources among a large number of users helps in reducing infrastructure costs and raising load capacities. Cluster Computing A cluster is a group of computers connected by a network. The group works together to accomplish the same task. Clusters are used mainly for load balancing. They shift processes between nodes to keep an even load on the group of connected computers. Hadoop architecture uses the similar methods (Chapter 2). Table 1.3 gives a comparison of grid computing

| Distributed computing | Cluster computing | Grid computing |
|---|---|---|
| • Loosely coupled<br>• Heterogeneous<br>• Single administration | • Tightly coupled<br>• Homogeneous<br>• Cooperative working | • Large scale<br>• Cross organizational<br>• Geographical distribution<br>• Distributed management |

### 1.3.5 Volunteer Computing

Volunteers provide computing resources to projects of importance that use resources to do distributed computing and/or storage. Volunteer computing is a distributed computing paradigm which uses computing resources of the volunteers. Volunteers are organizations or members who own personal computers. Projects examples are science-related projects

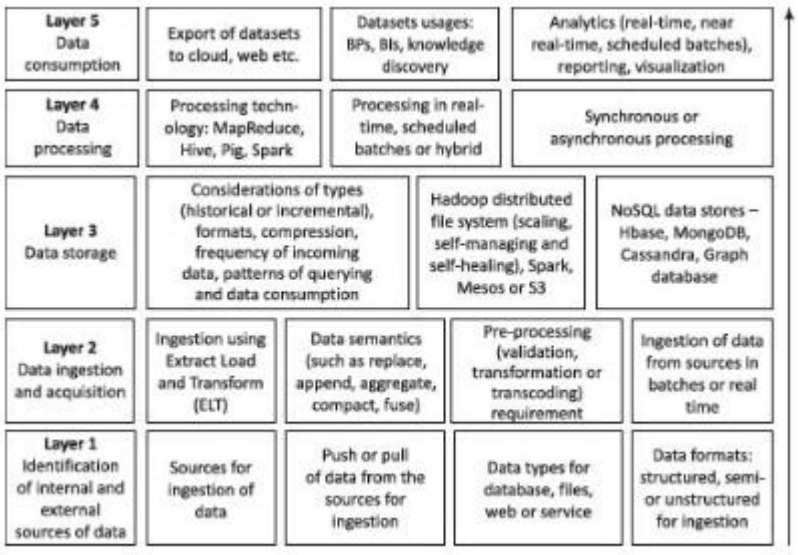
executed by universities or academia in general. Some issues with volunteer computing systems are:

1. Volunteered computers heterogeneity
2. Drop outs from the network over time
3. Their sporadic availability
4. Incorrect results at volunteers are unaccountable unaccountable as they are essentially from anonymous volunteers.

### 1.4 Designing Data Architecture:

#### 1.4.1 Data Architecture Design

Techopedia defines Big Data architecture as follows: "Big Data architecture is the logical and/or physical layout/structure of how Big Data will be stored, accessed and managed within a Big Data or IT environment. Architecture logically defines how Big Data solution will work, the core components (hardware, database, software, storage) used, flow of information, security and more." Characteristics of Big Data make designing Big Data architecture a complex process. Further, faster additions of new technological innovations increase the complexity in design. The requirements for offering competing products at lower costs in the market make the designing task more challenging for a Big Data architect. Data analytics need the number of sequential steps. Big Data architecture design task simplifies when using the logical layers approach. Figure 1.2 shows the logical layers and the functions which are considered in Big Data architecture. Five vertically aligned textboxes on the left of Figure 1.2 show the layers. Horizontal textboxes show the functions in each layer. Data processing architecture consists of five layers: (i) identification of data sources, (ii) acquisition, ingestion, extraction, pre-processing, transformation of data, (iii) data storage at files, servers, cluster or cloud, (iv) data-processing, and (v) data consumption in the number of programs and tools.



Data consumed for applications, such as business intelligence, data mining, discovering patterns/clusters, artificial intelligence (AI), machine learning (ML), text analytics, descriptive and predictive analytics, and data visualization. Data ingestion, pre-processing, storage and analytics require special tools and technologies. Logical layer 1 (L1) is for identifying data sources, which are external, internal or both. The layer 2 (L2) is for data-ingestion. Data ingestion means a process of absorbing information, just like the process of absorbing nutrients and medications into the body by eating or drinking them (Cambridge English Dictionary). Ingestion is the process of obtaining and importing data for immediate use or transfer. Ingestion may be in batches or in real time using pre-processing or semantics. The L3 layer is for storage of data from the L2 layer. The L4 is for data processing using software, such as MapReduce, Hive, Pig or Spark. The top layer L5 is for data consumption. Data is used in analytics, visualizations, reporting, export to cloud or web servers.

L1 considers the following aspects in a design:

- Amount of data needed at ingestion layer 2 (L2)
- Push from L1 or pull by L2 as per the mechanism for the usages
- Source data-types: Database, files, web or service
- Source formats, i.e., semi-structured, unstructured or structured.

L2 considers the following aspects:

- Ingestion and ETL processes either in real time, which means store and use the data as generated, or in batches. Batch processing is using discrete datasets at scheduled or periodic intervals of time.

L3 considers the followings aspects:

- Data storage type (historical or incremental), format, compression, incoming data frequency, querying patterns and consumption requirements for L4 or L5
- Data storage using Hadoop distributed file system or NoSQL data stores—HBase, Cassandra, MongoDB.

L4 considers the followings aspects:

- Data processing software such as MapReduce, Hive, Pig, Spark, Spark Mahout, Spark Streaming
- Processing in scheduled batches or real time or hybrid
- Processing as per synchronous or asynchronous processing requirements at L5.

L5 considers the consumption of data for the following:

- Data integration
- Datasets usages for reporting and visualization
- Analytics (real time, near real time, scheduled batches), BPs, BIs, knowledge discovery
- Export of datasets to cloud, web or other system

### 1.4.2 Managing Data for Analysis

Data managing means enabling, controlling, protecting, delivering and enhancing the value of data and information asset. Reports, analysis and visualizations need well-defined data. Data management also enables data usage in applications. The process for managing needs to be well defined for fulfilling requirements of the applications.

Data management functions include:

- Data assets creation, maintenance and protection
- Data governance, which includes establishing the processes for ensuring the availability, usability, integrity, security and high-quality of data. The processes enable trustworthy data availability for analytics, followed by the decision making at the enterprise.
- Data architecture creation, modelling and analysis
- Database maintenance, administration and management system. For example, RDBMS (relational database management system), NoSQL
- Managing data security, data access control, deletion, privacy and security
- Managing the data quality
- Data collection using the ETL process
- Managing documents, records and contents
- Creation of reference and master data, and data control and supervision
- Data and application integration
- Integrated data management, enterprise-ready data creation, fast access and analysis, automation and simplification of operations on the data,
- Data warehouse management
- Maintenance of business intelligence
- Data mining and analytics algorithms.

### 1.6.1 Data Storage and Management: Traditional Systems

1.6.1.1 Data Store with Structured or Semi-Structured Data Traditional systems use structured or semi-structured data. The following example explains the sources and data store of structured data.

1.6.1.2

SQL An RDBMS uses SQL (Structured Query Language). SQL is a language for viewing or changing (update, insert or append or delete) databases. It is a language for data access control, schema creation and data modifications. SQL was originally based on the tuple relational calculus and relational algebra. SQL can embed within other languages using SQL modules, libraries and pre-compilers. SQL does the following:

1. Create schema, which is a structure which contains description of objects (base tables, views, constraints) created by a user. The user can describe the data and define the data in the database.
2. Create catalog, which consists of a set of schemas which describe the database.
3. Data Definition Language (DDL) for the commands which depicts a database, that include creating, altering and dropping of tables and establishing the constraints. A user

can create and drop databases and tables, establish foreign keys, create view, stored procedure, functions in the database etc.

4. Data Manipulation Language (DML) for commands that maintain and query the database. A user can manipulate (INSERT/UPDATE) and access (SELECT) the data.
5. Data Control Language (DCL) for commands that control a database, and include administering of privileges and committing. A user can set (grant, add or revoke) permissions on tables, procedures and views.

SQL is a language for managing the RDBMS. A relational DB is a collection of data in multiple tables, which relate to each other through special fields, called keys (primary key, foreign key and unique key). Relational databases provide flexibilities. Relational database examples are MySQL PostGreSQL Oracle database, Informix, IBM DB2 and Microsoft SQL server.

### 1.6.1.3 Large Data Storage using RDBMS

RDBMS tables store data in a structured form. The tables have rows and columns. Data management of Data Store includes the provisions for privacy and security, data integration, compaction and fusion. The systems use machine-generated data, human-sourced data, and data from business processes (BP) and business intelligence (BI). A set of keys and relational keys access the fields

at tables, and retrieve data using queries (insert, modify, append, join or delete). RDBMSs use software for data administration also. Online content associated with Practice Exercise 1.12 describes the use of tables in relational databases in detail.

### 1.6.1.4 Distributed Database Management System

A distributed DBMS (DDBMS) is a collection of logically interrelated databases at multiple system over a computer network. The features of a distributed database system are:

1. A collection of logically related databases.
2. Cooperation between databases in a transparent manner. Transparent means that each user within the system may access all of the data within all of the databases as if they were a single database.
3. Should be ‘location independent’ which means the user is unaware of where the data is located, and it is possible to move the data from one physical location to another without affecting the user.

### 1.6.1.5 In-Memory Column Formats Data

A columnar format in-memory allows faster data retrieval when only a few columns in a table need to be selected during query processing or aggregation. Data in a column are kept together in-memory in columnar format. A single memory access, therefore, loads many values at the column. An address increment to a next memory address for the next value is fast when compared to first computing the address of the next value, which is not the immediate next address. The following example explains the in-memory columnar format.

Online Analytical Processing (OLAP) in real-time transaction processing is fast when using in-memory column format tables. OLAP enables real-time analytics. The CPU accesses all columns in a single instance of access to the memory in columnar format in-memory data-storage.

Online Analytical Processing (OLAP) enables online viewing of analyzed data and visualization up to the desired granularity (fineness or coarseness) enables view by rolling up (finer granulates to coarse granulates data) or drilling down (coarser granulates data to finer granulates). OLAP enables obtaining online summarized information and automated reports for a large database. Metadata describes the data. Pre-storing of calculated values provide consistently fast response. Result formats from the queries are based on Metadata.

### 1.6.1.6 In-Memory Row Format Databases

A row format in-memory allows much faster data processing during OLTP (online transaction processing). Refer Example 1.13. Each row record has corresponding values in multiple columns and the on-line values store at the consecutive memory addresses in row format. A specific day's sale of five different chocolate flavours is stored in consecutive columns c to c+5 at memory. A single instance of memory accesses loads values of all five flavours at successive columns during online processing. For example, the total number of chocolates sold computes online. Data is in-memory row-formats in stream and event analytics. The stream analytics method does continuous computation that happens as data is flowing through the system. Event analytics does computation on event and use event data for tracking and reporting events.

### 1.6.1.7 Enterprise Data-Store Server and Data Warehouse

Enterprise data, after data cleaning process, integrate with the server data at warehouse. Enterprise data server use data from several distributed sources which store data using various technologies. All data merge using an integration tool. Integration enables collective viewing of the datasets at the data warehouse (Figure 1.3). Enterprise data integration may also include integration with application(s), such as analytics, visualization, reporting, business intelligence and knowledge discovery. Heterogeneous systems execute complex integration processes when integrating at an enterprise server or data warehouse. Complex application-integration means the integration of heterogeneous application architectures and processes with the databases at the enterprise. Enterprise data warehouse store the databases, and data stores after integration, using tools from number of sources. Online contents associated with Practice Exercises 1.9 and 1.10 give details of commercial solutions for complex application-integration of processes. Following are some standardised business processes, as defined in the Oracle application-integration architecture:

1. Integrating and enhancing the existing systems and processes
2. Business intelligence
3. Data security and integrity
4. New business services/products (Web services)
5. Collaboration/knowledge management
6. Enterprise architecture/SOA

7. e-commerce
8. External customer services
9. Supply chain automation/visualization
10. Data centre optimization

### 1.6.2 Big Data Storage

Following subsections describe Big Data storage concepts:

#### 1.6.2.1 Big Data NoSQL or Not Only SQL

NoSQL databases are considered as semi-structured data. Big Data Store uses NoSQL. NOSQL stands for No SQL or Not Only SQL. The stores do not integrate with applications using SQL. NoSQL is also used in cloud data store. Features of NoSQL are as follows: It is a class of non-relational data storage systems, and the flexible data models and multiple schema:

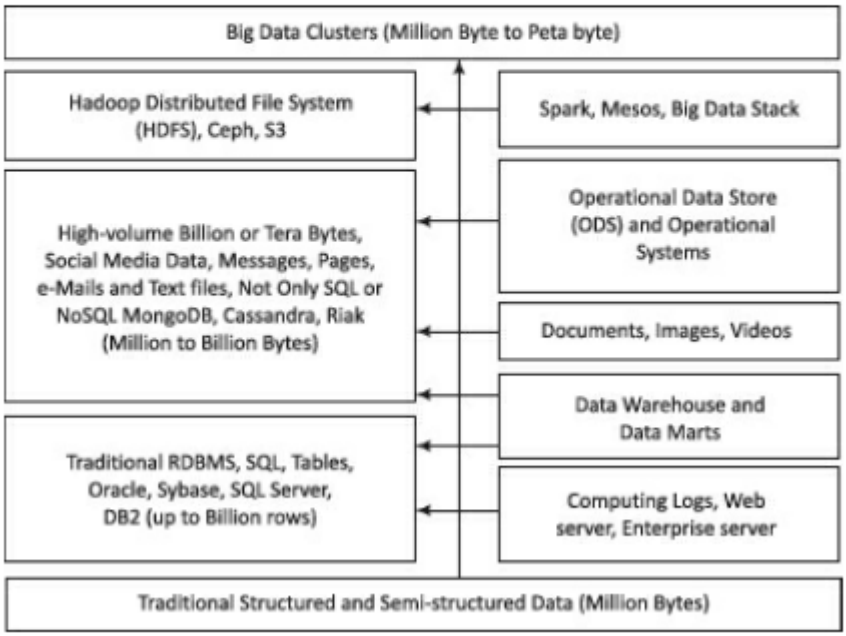
(i) Class consisting of uninterrupted key/value or big hash table [Dynamo (Amazon S3)]
(ii) Class consisting of unordered keys and using JSON (PNUTS)
(iii) Class consisting of ordered keys and semi-structured data storage systems [BigTable, Cassandra (used in Facebook/Apache) and HBase]
(iv) Class consisting of JSON (MongoDB)
(v) Class consisting of name/value in the text (CouchDB)
(vi) May not use fixed table schema
(vii) (vii) Do not use the JOINS
(viii) (viii) Data written at one node can replicate at multiple nodes, therefore Data storage is fault-tolerant,
(ix) (ix) May relax the ACID rules during the Data Store transactions.
(x) (x) Data Store can be partitioned and follows CAP theorem (out of three properties, consistency, availability and partitions, at least two must be there during the transactions) Consistency means all copies have the same valuelike in traditional DBs. Availability means at least one copy is available in case a partition becomes inactive or fails. For example in web applications, the other copy in other partition is available. Partition means parts which are active but may not cooperate as in the distributed DBs. 1.6.2.2 Coexistence of Big Data, NoSQL and Traditional Data Stores Figure 1.7 shows co-existence of data at server, SQL, RDBMS with NoSQL and Big Data at Hadoop, Spark, Mesos, S3 or compatible Clusters. Table 1.4 gives various data sources for Big Data along with its examples of usages and the tools

| Data Source | Examples of Usages | Example of Tools |
|---|---|---|
| Relational databases | Managing business applications involving structured data | Microsoft Access, Oracle, IBM DB2, SQL Server, MySQL, PostgreSQL Composite, SQL on Hadoop [HPE (Hewlett Packard Enterprise) Vertica, IBM BigSQL, Microsoft Polybase, Oracle Big Data SQL] |
| Analysis databases (MPP, columnar, In-memory) | High performance queries and analytics | Sybase IQ, Kognitio, Terradata, Netezza, Vertica, ParAccel, ParStream, Infobright, Vectorwise, |
| NoSQL databases | Key-value pairs, fast read/ | Key-value pair |

| | | |
|---|---|---|
| (Key-value pairs, Columnar format, documents, Objects, graph) | write using collections of name-value pairs for storing any type of data; Columnar format, documents, objects, graph DBs and DSs | databases: Riak DS (Data Store), OrientDB, Column format databases (HBase, Cassandra), Document oriented databases: CouchDB, MongoDB; Graph databases (Neo4j, Tetan) |
| Hadoop clusters | Ability to process large data sets across a distributed computing environment | Cloudera, Apache HDFS |
| Web applications | Access to data generated from web applications | Google Analytics, Twitter |
| Cloud data | Elastic scalable outsourced databases, and data administration services | Amazon Web Services, Rackspace, GoogleSQL |
| Individual data | Individual productivity | MS Excel, CSV, TLV, JSON, MIME type |
| Multidimensional | Well-defined bounded exploration especially | Microsoft SQL Server Analysis Services |

Hadoop, Spark and compatible Big Data Clusters 1.6.3 Big Data Platform A Big Data platform supports large datasets and volume of data. The data generate at a higher velocity, in more varieties or in higher veracity. Managing Big Data requires large resources of MPPs, cloud, parallel processing and specialized tools. Bigdata platform should provision tools and services for: storage, processing and analytics, developing, deploying, operating and

| | popular for financial applications | |
|---|---|---|
| Social media data | Text data, images, videos | Twitter, LinkedIn |




### 1.6.3 Big Data Platform

A Big Data platform supports large datasets and volume of data. The data generate at a higher velocity, in more varieties or in higher veracity. Managing Big Data requires large resources of MPPs, cloud, parallel processing and specialized tools. Bigdata platform should provision tools and services for:

1. storage, processing and analytics,
2. developing, deploying, operating and managing Big Data environment,
3. reducing the complexity of multiple data sources and integration of applications into one cohesive solution,
4. custom development, querying and integration with other systems, and
5. the traditional as well as Big Data techniques.

Data management, storage and analytics of Big data captured at the companies and services require the following:

1. New innovative non-traditional methods of storage, processing and analytics
2. Distributed Data Stores
3. Creating scalable as well as elastic virtualized platform (cloud computing)
4. Huge volume of Data Stores
5. Massive parallelism

6. High speed networks
7. High performance processing, optimization and tuning
8. Data management model based on Not Only SQL or NoSQL
9. In-memory data column-formats transactions processing or dual in-memory data columns as well as row formats for OLAP and OLTP
10. Data retrieval, mining, reporting, visualization and analytics
11. Graph databases to enable analytics with social network messages, pages and data analytics Machine learning or other approaches Big data sources: Data storages, data warehouse, Oracle Big Data, MongoDB
12. NoSQL, Cassandra NoSQL Data sources: Sensors, Audit trail of Financial transactions data, external data such as Web, Social Media, weather data, health records data.

### 1.6.3.1 Hadoop

Big Data platform consists of Big Data storage(s), server(s) and data management and business intelligence software. Storage can deploy Hadoop Distributed File System (HDFS), NoSQL data stores, such as HBase, MongoDB, Cassandra. HDFS system is an open source storage system. HDFS is a scaling, self-managing and self-healing file system. The Hadoop system packages application-programming model. Hadoop is a scalable and reliable parallel computing platform. Hadoop manages Big Data distributed databases. Figure 1.8 shows Hadoop based Big Data environment. Small height cylinders represent MapReduce and big ones represent the Hadoop. 1.6.3.2 Mesos **Mesos** v0.9 is a resources management platform which enables sharing of cluster of nodes by multiple frameworks and which has compatibility with an open analytics stack [data processing (Hive, Hadoop, HBase, Storm), data management (HDFS)].

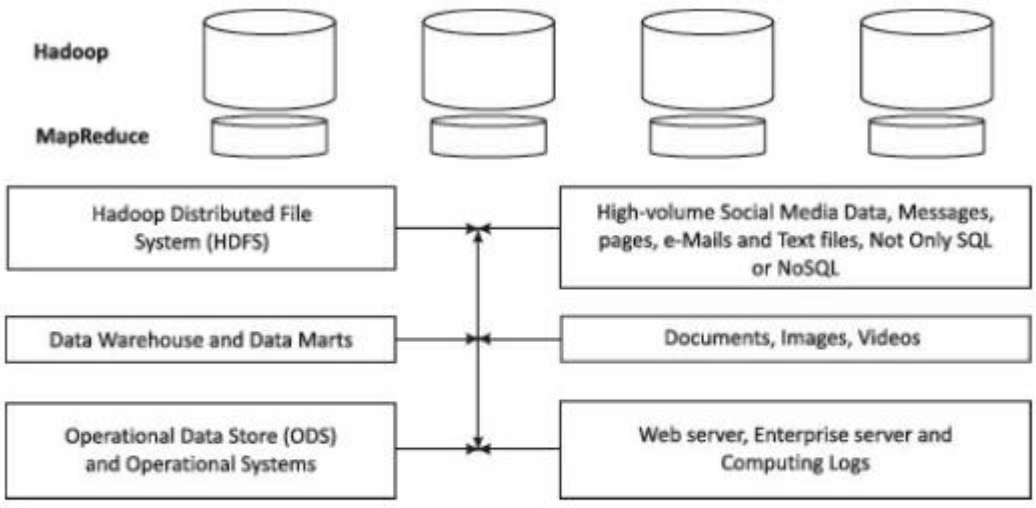



**Figure 1.8** Hadoop based Big Data environment

### 1.6.3.3 Big Data Stack

A stack consists of a set of software components and data store units. Applications, machine-learning algorithms, analytics and visualization tools use Big Data Stack (BDS) at a cloud service, such as Amazon EC2, Azure or private cloud. The stack uses cluster of high performance machines. Table 1.5 gives Big Data management, storage and processing tools.

Table 1.5 Tools for Big Data environment

| Types | Examples |
|---|---|
| MapReduce | Hadoop, Apache Hive, Apache Pig, Cascading, Cascalog, mrjob (Python MapReduce library), Apache S4, MapR, Apple Acunu, Apache Flume, Apache Kafka |
| NoSQL Databases | MongoDB, Apache CouchDB, Apache Cassandra, Aerospike, Apache HBase, Hypertable |
| Processing | Spark, IBM BigSheets, PySpark, R, Yahoo! Pipes, Amazon Mechanical Turk, Datameer, Apache Solr/Lucene, ElasticSearch |
| Servers | Amazon EC2, S3, GoogleQuery, Google App Engine, AWS Elastic Beanstalk, Salesforce Heroku |
| Storage | Hadoop Distributed File System, Amazon |

### 1.6.4 Big Data Analytics

DBMS or RDBMS manages the traditional databases. Data analysis need pre-processing of raw data and gives information useful for decision making. Analysis brings order, structure and

meaning to the collection of data. Data is collected and analyzed to answer questions, test the hypotheses or disprove theories.

### 1.6.4.1 Data Analytics Definition

Data Analytics can be formally defined as the statistical prescriptive capability. Analytics uses historical data and forecasts new values or results. Analytics suggests techniques which will provide the most efficient and beneficial results for an enterprise. Data analysis helps in finding business intelligence and helps in decision making. Data analysis can be defined as, "Analysis of data is a process of inspecting, cleaning, transforming and modeling data with the goal of discovering useful information, suggesting conclusions and supporting decision making." (Wikipedia)

### 1.6.4.2 Phases in Analytics

Analytics has the following phases before deriving the new facts, providing business intelligence and generating new knowledge.

1. Descriptive analytics enables deriving the additional value from visualizations and reports
2. Predictive analytics is advanced analytics which enables extraction of new facts and knowledge, and then predicts/forecasts
3. Prescriptive analytics enable derivation of the additional value and undertake better decisions for new option(s) to maximize the profits
4. Cognitive analytics enables derivation of the additional value and undertake better decisions.

Analytics integrates with the enterprise server or data warehouse. Figure 1.9 shows an overview of a reference model for analytics architecture. The figure also shows on the right-hand side the Big Data file systems, machine learning algorithms and query languages and usage of the Hadoop ecosystem.

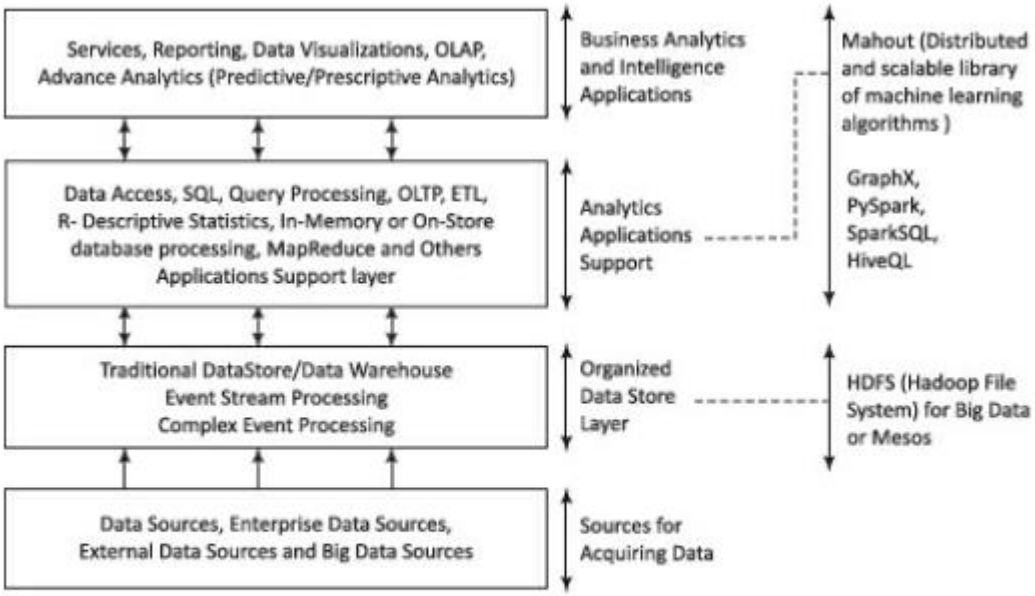



**Figure 1.9** Traditional and Big Data analytics architecture reference model

The captured or stored data require a well-proven strategy to calculate, plan or analyze. When Big Data combine with high-powered data analysis, enterprise achieve valued business-related tasks. Examples are: Determine root causes of defects, faults and failures in minimum time. Deliver advertisements on mobiles or web, based on customer's location and buying habits. Detect offender before that affects the organization or society.

### 1.6.4.3 Berkeley Data Analytics Stack (BDAS)

The importance of Big Data lies in the fact that what one does with it rather than how big or large it is. Identify whether the gathered data is able to help in obtaining the following findings: 1) cost reduction, 2) time reduction, 3) new product planning and development, 4) smart decision making using predictive analytics and 5) knowledge discovery. Big Data analytics need innovative as well as cost effective techniques. BDAS is an open-source data analytics stack for complex computations on Big Data.11 It supports efficient, large-scale in-memory data processing, and thus enables user applications achieving three fundamental processing requirements; accuracy, time and cost. Berkeley Data Analytics Stack (BDAS) consists of data processing, data management and resource management layers. Following list these: Applications, AMP-Genomics and Carat run at the BDAS. Data processing software component provides in-memory processing which processes the data efficiently across the frameworks. AMP stands for Berkeley's Algorithms, Machines and Peoples Laboratory. Data processing combines batch, streaming and interactive computations.

Resource management software component provides for sharing the infrastructure across various frameworks. Figure 1.10 shows a four layers architecture for Big Data Stack that

consists of Hadoop, MapReduce, Spark core and SparkSQL, Streaming, R, GraphX, MLib, Mahout, Arrow and Kafka.