CHAPTER 4

The Big Data Technology Landscape

BRIEF CONTENTS

- What's in Store?
- NoSQL (Not Only SQL)
  - Where is it used?
  - What is it?
  - Types of NoSQL Databases
  - Why NoSQL?
  - Advantages of NoSQL
  - What we miss with NoSQL?
  - Use of NoSQL in Industry
  - NoSQL Vendors
  - SQL versus NoSQL
  - NewSQL
  - Comparison of SQL, NoSQL, and NewSQL
- Hadoop
  - Features of Hadoop
  - Key Advantages of Hadoop
  - Versions of Hadoop
    - Hadoop 1.0
    - Hadoop 2.0
- Overview of Hadoop Ecosystems
- Hadoop Distributions
- Hadoop versus SQL
- Integrated Hadoop Systems Offered by Leading Market Vendors
- Cloud-Based Hadoop Solutions

"The goal is to turn data into information, and information into insight."

– Carly Fiorina, former CEO, Hewlett-Packard Co

WHAT'S IN STORE?

The focus of this chapter is on understanding "big data technology landscape". This chapter is an overview on NoSQL and Hadoop. There are separate chapters on NoSQL (MongoDB and Cassandra) as well as Hadoop in the book.

The big data technology landscape can be majorly studied under two important technologies:

1. NoSQL
2. Hadoop

4.1 NoSQL (NOT ONLY SQL)

The term NoSQL was first coined by Carlo Strozzi in 1998 to name his lightweight, open-source, non-relational database that did not expose the standard SQL interface. The term was reintroduced by Eric Evans in early 2009.

4.1.1 Where is it Used?

NoSQL databases are widely used in big data and other real-time web applications. Refer Figure 4.1. NoSQL databases is used to stock log data which can then be pulled for analysis. Likewise it is used to store social media data and all such data which cannot be stored and analyzed comfortably in RDBMS.

4.1.2 What is it?

NoSQL stands for Not Only SQL. These are non-relational, open source, distributed databases. They are hugely popular today owing to their ability to scale out or scale horizontally and the adeptness at dealing with a rich variety of data: structured, semi-structured and unstructured data. Refer Figure 4.2 for additional features of NoSQL.

1. **NoSQL databases are non-relational:** They do not adhere to relational data model. In fact, they are either key-value pairs or document-oriented or column-oriented or graph-based databases.
2. **Distributed:** They are distributed meaning the data is distributed across several nodes in a cluster constituted of low-cost commodity hardware.

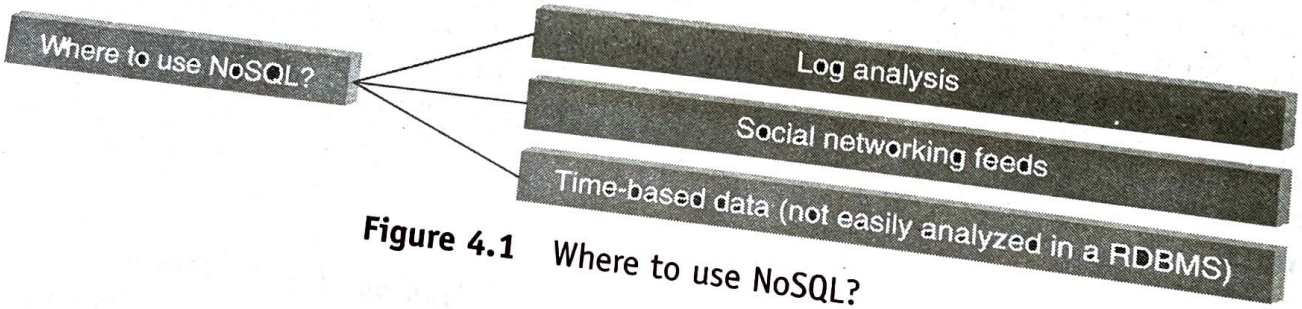


Figure 4.1 Where to use NoSQL?

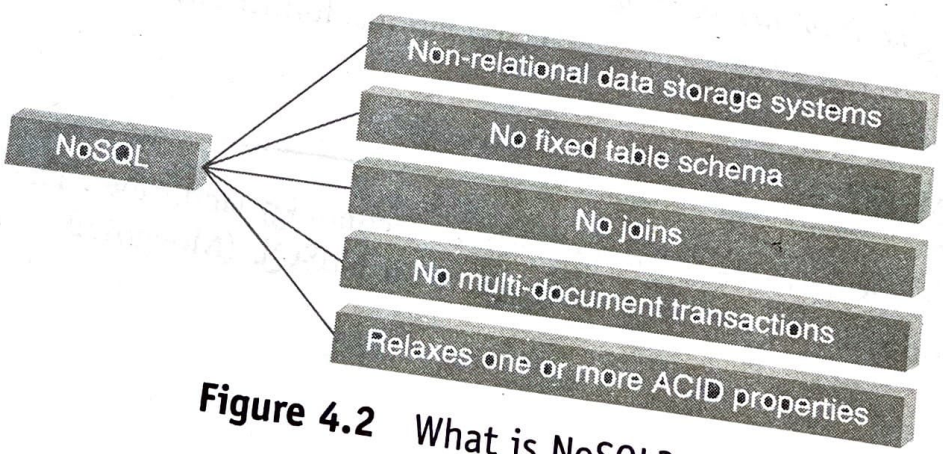


Figure 4.2 What is NoSQL?

3. **No support for ACID properties (Atomicity, Consistency, Isolation, and Durability):** They do not offer support for ACID properties of transactions. On the contrary, they have adherence to Brewer's CAP (Consistency, Availability, and Partition tolerance) theorem and are often seen compromising on consistency in favor of availability and partition tolerance.
4. **No fixed table schema:** NoSQL databases are becoming increasing popular owing to their support for flexibility to the schema. They do not mandate for the data to strictly adhere to any schema structure at the time of storage.

4.1.3 Types of NoSQL Databases

We have already stated that NoSQL databases are non-relational. They can be broadly classified into the following:

1. Key-value or the big hash table.
2. Schema-less.

Refer Figure 4.3. Let us take a closer look at key-value and few other types of schema-less databases:

1. **Key-value:** It maintains a big hash table of keys and values. For example, Dynamo, Redis, Riak, etc.
   Sample Key-Value Pair in Key-Value Database

Key	Value
First Name	Simmonds
Last Name	David

2. **Document:** It maintains data in collections constituted of documents. For example, MongoDB, Apache CouchDB, Couchbase, MarkLogic, etc.
   Sample Document in Document Database

```
{
"Book Name":    "Fundamentals of Business Analytics",
"Publisher":    "Wiley India",
"Year of Publication":    "2011"
}
```

3. **Column:** Each storage block has data from only one column. For example: Cassandra, HBase, etc.

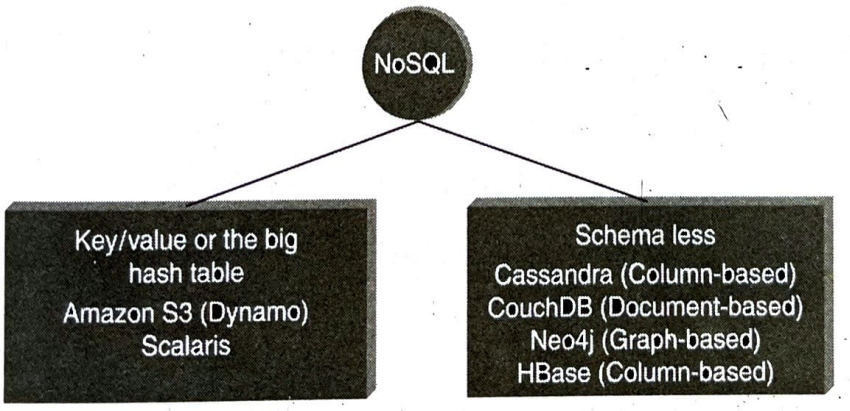
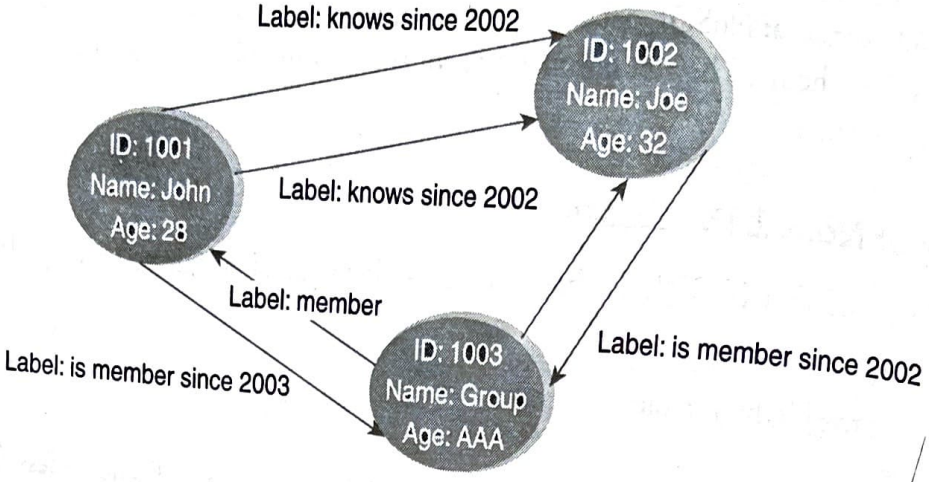


Figure 4.3 Types of NoSQL databases.

4. **Graph:** They are also called network database. A graph stores data in nodes. For example, Neo HyperGraphDB, etc.

Sample Graph in Graph Database



Refer Table 4.1 for popular schema-less databases.

4.1.4 Why NoSQL?

1. It has scale out architecture instead of the monolithic architecture of relational databases.
2. It can house large volumes of structured, semi-structured, and unstructured data.
3. **Dynamic schema:** NoSQL database allows insertion of data without a pre-defined schema. In other words, it facilitates application changes in real time, which thus supports faster development, easy code integration, and requires less database administration.
4. **Auto-sharding:** It automatically spreads data across an arbitrary number of servers. The application in question is more often not even aware of the composition of the server pool. It balances the load of data and query on the available servers; and if and when a server goes down, it is quickly replaced without any major activity disruptions.
5. **Replication:** It offers good support for replication which in turn guarantees high availability, fault tolerance, and disaster recovery.

4.1.5 Advantages of NoSQL

Let us enumerate the advantages of NoSQL. Refer Figure 4.4.

1. **Can easily scale up and down:** NoSQL database supports scaling rapidly and elastically and even allows to scale to the cloud.

Table 4.1 Popular schema-less databases

Key–Value Data Store	Column-Oriented Data Store	Document Data Store	Graph Data Store
• Riak	• Cassandra	• MongoDB	• InfiniteGraph
• Redis	• HBase	• CouchDB	• Neo4j
• Membase	• HyperTable	• RavenDB	• AllegroGraph

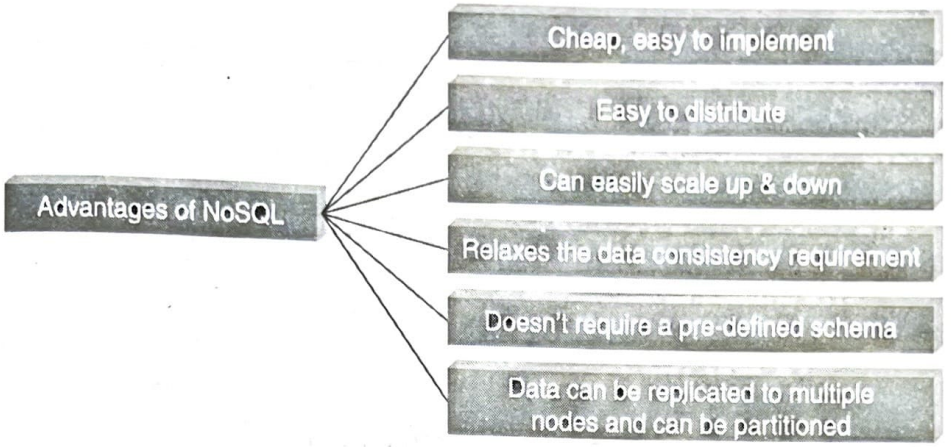


Figure 4.4 Advantages of NoSQL.

(a) ***Cluster scale:*** It allows distribution of database across 100+ nodes often in multiple data centers.
(b) ***Performance scale:*** It sustains over 100,000+ database reads and writes per second.
(c) ***Data scale:*** It supports housing of 1 billion+ documents in the database.

2. **Doesn't require a pre-defined schema:** NoSQL does not require any adherence to pre-defined schema. It is pretty flexible. For example, if we look at MongoDB, the documents (equivalent of records in RDBMS) in a collection (equivalent of table in RDBMS) can have different sets of key-value pairs.

   {_id: 101,"BookName": "Fundamentals of business analytics", "AuthorName": "Seema Acharya", "Publisher": "Wiley India"}
   {_id:102, "BookName":"Big Data and Analytics"}

3. **Cheap, easy to implement:** Deploying NoSQL properly allows for all of the benefits of scale, high availability, fault tolerance, etc. while also lowering operational costs.
4. **Relaxes the data consistency requirement:** NoSQL databases have adherence to CAP theorem (Consistency, Availability, and Partition Tolerance). Most of the NoSQL databases compromise on consistency in favor of availability and partition tolerance. However, they do go for eventual consistency.
5. **Data can be replicated to multiple nodes and can be partitioned:** There are two terms that we will discuss here:

   (a) ***Sharding:*** Sharding is when different pieces of data are distributed across multiple servers. NoSQL databases support auto-sharding meaning they can natively and automatically spread data across an arbitrary number of servers, without requiring the application to even be aware of the composition of the server pool. Servers can be added or removed from the data layer without application downtime. This would mean that data and query load are automatically balanced across servers, and when a server goes down, it can be quickly and transparently replaced with no application disruption.
   (b) ***Replication:*** Replication is when multiple copies of data are stored across the cluster and even across data centers. This promises high availability and fault tolerance.

4.1.6 What We Miss With NoSQL?

With NoSQL around, we have been able to counter the problem of scale (NoSQL scales out). There is also the flexibility with respect to schema design. However there are few features of conventional RDBMS that are greatly missed. Refer Figure 4.5.

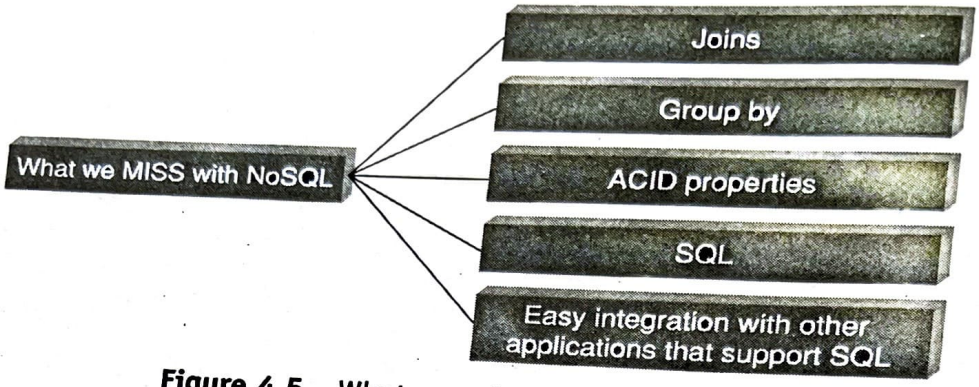


Figure 4.5 What we miss with NoSQL?

NoSQL does not support joins. However, it compensates for it by allowing embedded documents as in MongoDB. It does not have provision for ACID properties of transactions. However, it obeys the Eric Brewer's CAP theorem. NoSQL does not have a standard SQL interface but NoSQL databases such as MongoDB and Cassandra have their own rich query language [MongoDB query language and Cassandra query language (CQL)] to compensate for the lack of it. One thing which is dearly missed is the easy integration with other applications that support SQL.

4.1.7 Use of NoSQL in Industry

NoSQL is being put to use in varied industries. They are used to support analysis for applications such as web user data analysis, log analysis, sensor feed analysis, making recommendations for upsell and cross-sell, etc. Refer Figure 4.6.

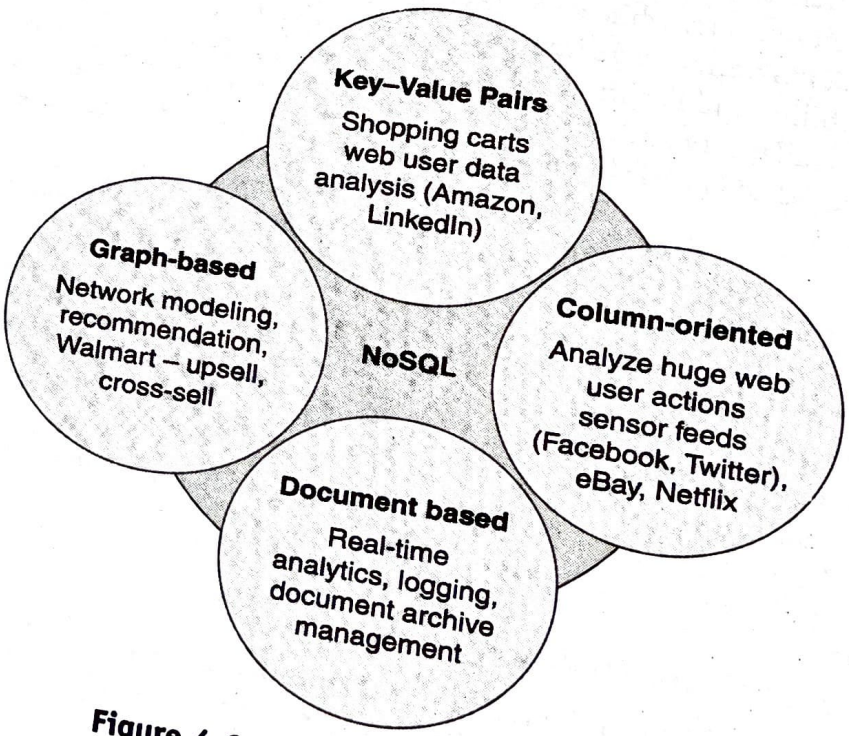


Figure 4.6 Use of NoSQL

4.1.8 NoSQL Vendors

Refer Table 4.2 for few popular NoSQL vendors.

4.1.9 SQL versus NoSQL

Refer Table 4.3 for few salient differences between SQL and NoSQL.

Table 4.2 Few popular NoSQL vendors

Company	Product	Most Widely Used by
Amazon	DynamoDB	LinkedIn, Mozilla
Facebook	Cassandra	Netflix, Twitter, eBay
Google	BigTable	Adobe Photoshop

Table 4.3 SQL versus NoSQL

SQL	NoSQL
Relational database	Non-relational, distributed database
Relational model	Model-less approach
Pre-defined schema	Dynamic schema for unstructured data
Table based databases	Document-based or graph-based or wide column store or key-value pairs databases
Vertically scalable (by increasing system resources)	Horizontally scalable (by creating a cluster of commodity machines)
Uses SQL	Uses UnQL (Unstructured Query Language)
Not preferred for large datasets	Largely preferred for large datasets
Not a best fit for hierarchical data	Best fit for hierarchical storage as it follows the key-value pair of storing data similar to JSON (Java Script Object Notation)
Emphasis on ACID properties	Follows Brewer's CAP theorem
Excellent support from vendors	Relies heavily on community support
Supports complex querying and data keeping needs	Does not have good support for complex querying
Can be configured for strong consistency	Few support strong consistency (e.g., MongoDB), few others can be configured for eventual consistency (e.g., Cassandra)
Examples: Oracle, DB2, MySQL, MS SQL, PostgreSQL, etc.	MongoDB, HBase, Cassandra, Redis, Neo4j, CouchDB, Couchbase, Riak, etc.

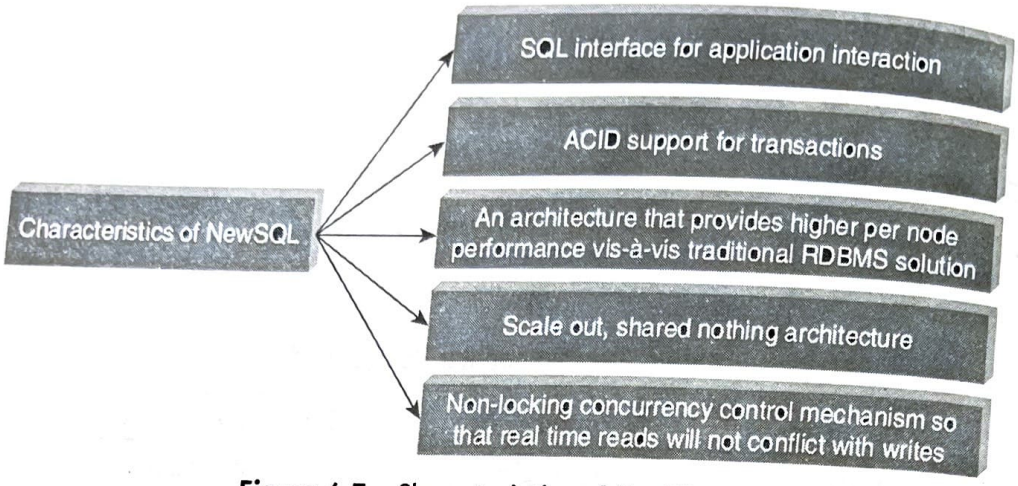


Figure 4.7 Characteristics of NewSQL.

4.1.10 NewSQL

There is yet another new term doing the rounds – "NewSQL". So, what is NewSQL and how is it different from SQL and NoSQL?

What is that we love about NoSQL and is not there with our traditional RDBMS and what is that we love about SQL that NoSQL does not have support for? You guessed it right!!! We need a database that has the same scalable performance of NoSQL systems for On Line Transaction Processing (OLTP) while still maintaining the ACID guarantees of a traditional database. This new modern RDBMS is called NewSQL. It supports relational data model and uses SQL as their primary interface.

4.1.10.1 Characteristics of NewSQL

Refer Figure 4.7 to learn about the characteristics of NewSQL. NewSQL is based on the shared nothing architecture with a SQL interface for application interaction.

4.1.11 Comparison of SQL, NoSQL, and NewSQL

Refer Table 4.4 for a comparative study of SQL, NoSQL and NewSQL.

Table 4.4 Comparative study of SQL, NoSQL, and NewSQL

	SQL	NoSQL	NewSQL
Adherence to ACID properties	Yes	No	Yes
OLTP/OLAP	Yes	No	Yes
Schema rigidity	Yes	No	Maybe
Adherence to data model	Adherence to relational model	Yes	Maybe
Data Format Flexibility	No	Scale out Horizontal Scaling	Scale out
Scalability	Scale up Vertical Scaling	Yes	Yes
Distributed Computing	Yes	Growing	Slowly growing
Community Support	Huge		

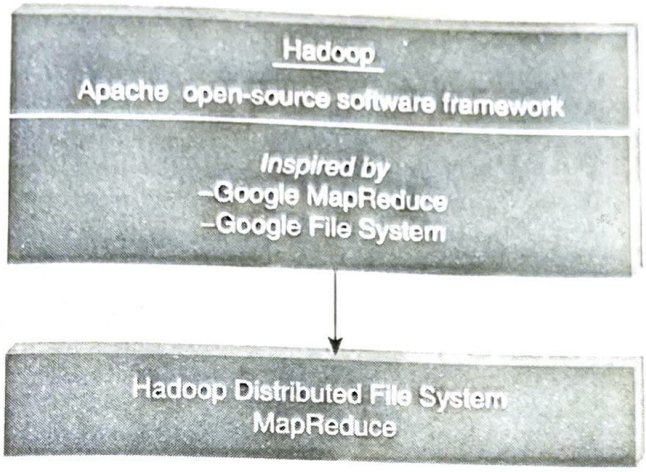


Figure 4.8 Hadoop.

4.2 HADOOP

Hadoop is an open-source project of the Apache foundation. It is a framework written in Java, originally developed by Doug Cutting in 2005 who named it after his son's toy elephant. He was working with Yahoo then. It was created to support distribution for "Nutch", the text search engine. Hadoop uses Google's MapReduce and Google File System technologies as its foundation. Hadoop is now a core part of the computing infrastructure for companies such as Yahoo, Facebook, LinkedIn and Twitter, etc. Refer Figure 4.8.

4.2.1 Features of Hadoop

Let us cite a few features of Hadoop:

1. It is optimized to handle massive quantities of structured, semi-structured, and unstructured data, using commodity hardware, that is, relatively inexpensive computers.
2. Hadoop has a shared nothing architecture.
3. It replicates its data across multiple computers so that if one goes down, the data can still be processed from another machine that stores its replica.
4. Hadoop is for high throughput rather than low latency. It is a batch operation handling massive quantities of data; therefore the response time is not immediate.
5. It complements On-Line Transaction Processing (OLTP) and On-Line Analytical Processing (OLAP). However, it is not a replacement for a relational database management system.
6. It is NOT good when work cannot be parallelized or when there are dependencies within the data.
7. It is NOT good for processing small files. It works best with huge data files and data sets.

4.2.2 Key Advantages of Hadoop

Refer Figure 4.9 for a quick look at the key advantages of Hadoop:

1. **Stores data in its native format:** Hadoop's data storage framework (HDFS – Hadoop Distributed File System) can store data in its native format. There is no structure that is imposed while keying in data or storing data. HDFS is pretty much schema-less. It is only later when the data needs to be processed that structure is imposed on the raw data.

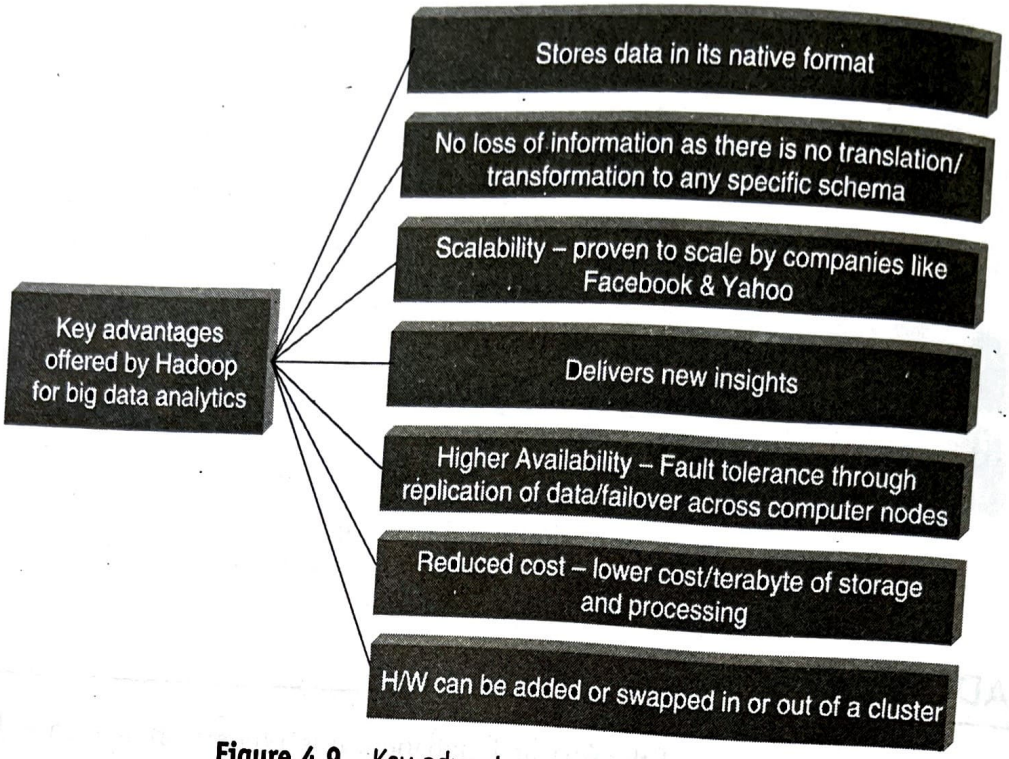


Figure 4.9 Key advantages of Hadoop.

2. **Scalable:** Hadoop can store and distribute very large datasets (involving thousands of terabytes of data) across hundreds of inexpensive servers that operate in parallel.
3. **Cost-effective:** Owing to its scale-out architecture, Hadoop has a much reduced cost/terabyte of storage and processing.
4. **Resilient to failure:** Hadoop is fault-tolerant. It practices replication of data diligently which means whenever data is sent to any node, the same data also gets replicated to other nodes in the cluster, thereby ensuring that in the event of a node failure, there will always be another copy of data available for use.
5. **Flexibility:** One of the key advantages of Hadoop is its ability to work with all kinds of data: structured, semi-structured, and unstructured data. It can help derive meaningful business insights from email conversations, social media data, click-stream data, etc. It can be put to several purposes such as log analysis, data mining, recommendation systems, market campaign analysis, etc.
6. **Fast:** the processing is extremely fast in Hadoop compared to other conventional systems owing to the "move code to data" paradigm.

4.2.3 Versions of Hadoop

There are two versions of Hadoop available:

1. Hadoop 1.0
2. Hadoop 2.0

Let us take a look at the features of both. Refer Figure 4.10.

4.2.3.1 Hadoop 1.0

It has two main parts:

1. **Data storage framework:** It is a general-purpose file system called Hadoop Distributed File System (HDFS). HDFS is schema-less. It simply stores data files and these data files can be in just about any

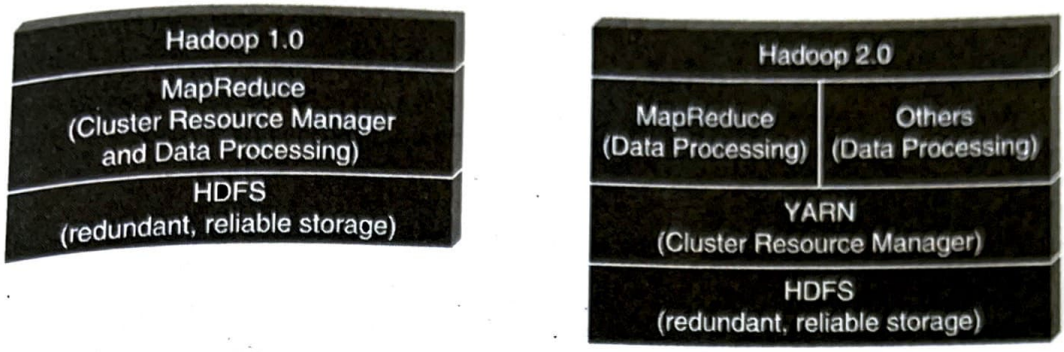


Figure 4.10 Versions of Hadoop.

format. The idea is to store files as close to their original form as possible. This is turn provides the business units and the organization the much needed flexibility and agility without being overly worried by what it can implement.

2. **Data processing framework:** This is a simple functional programming model initially popularized by Google as MapReduce. It essentially uses two functions: the MAP and the REDUCE functions to process data. The "Mappers" take in a set of key-value pairs and generate intermediate data (which is another list of key-value pairs). The "Reducers" then act on this input to produce the output data. The two functions seemingly work in isolation from one another, thus enabling the processing to be highly distributed in a highly-parallel, fault-tolerant, and scalable way.

There were however a few limitations of Hadoop 1.0. They are as follows:

1. The first limitation was the requirement for MapReduce programming expertise along with proficiency required in other programming languages, notably Java.
2. It supported only batch processing which although is suitable for tasks such as log analysis, large-scale data mining projects but pretty much unsuitable for other kinds of projects.
3. One major limitation was that Hadoop 1.0 was tightly computationally coupled with MapReduce, which meant that the established data management vendors were left with two options: Either rewrite their functionality in MapReduce so that it could be executed in Hadoop or extract the data from HDFS and process it outside of Hadoop. None of the options were viable as it led to process inefficiencies caused by the data being moved in and out of the Hadoop cluster.

Let us look at whether these limitations have been wholly or in parts resolved by Hadoop 2.0.

4.2.3.2 Hadoop 2.0

In Hadoop 2.0, HDFS continues to be the data storage framework. However, a new and separate resource management framework called **Y**et **A**nother **R**esource **N**egotiator (YARN) has been added. Any application capable of dividing itself into parallel tasks is supported by YARN. YARN coordinates the allocation of subtasks of the submitted application, thereby further enhancing the flexibility, scalability, and efficiency of the applications. It works by having an ApplicationMaster in place of the erstwhile JobTracker, running applications on resources governed by a new NodeManager (in place of the erstwhile TaskTracker). ApplicationMaster is able to run any application and not just MapReduce.

This, in other words, means that the MapReduce Programming expertise is no longer required. Furthermore, it not only supports batch processing but also real-time processing. MapReduce is no longer the only data processing option; other alternative data processing functions such as data standardization, master data management can now be performed natively in HDFS.

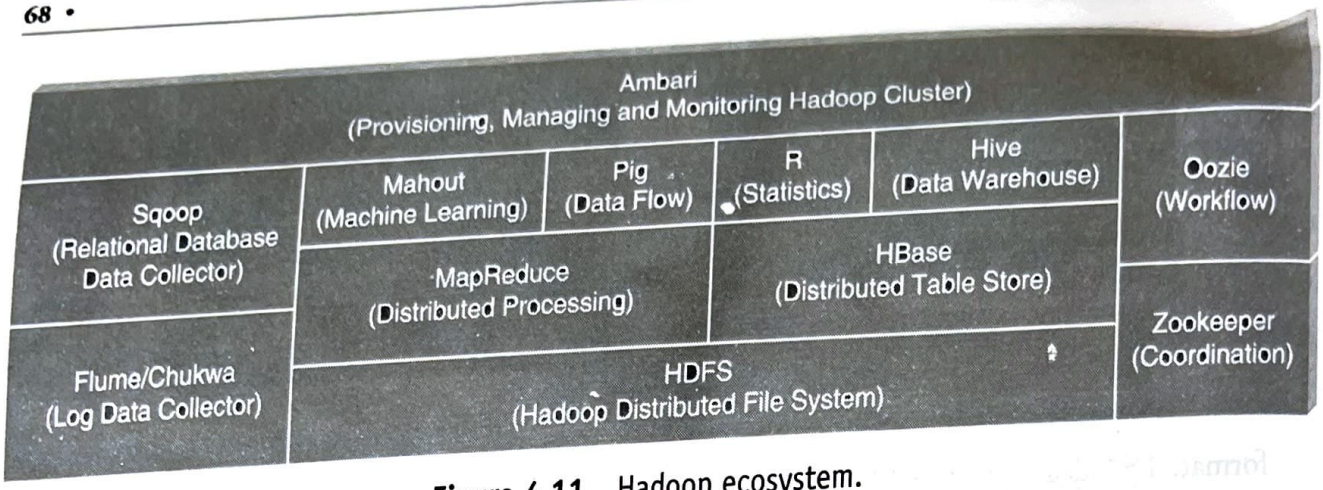


Figure 4.11 Hadoop ecosystem.

4.2.4 Overview of Hadoop Ecosystems

We will discuss the Hadoop ecosystem in brief here. It will be covered in detail in Chapter 5. The following are the components of the Hadoop ecosystem (shown in Figure 4.11):

1. **HDFS:** Hadoop Distributed File System. It simply stores data files as close to the original form as possible.
2. **HBase:** It is Hadoop's database and compares well with an RDBMS. It supports structured data storage for large tables.
3. **Hive:** It enables analysis of large data sets using a language very similar to standard ANSI SQL. This implies that anyone familiar with SQL should be able to access data stored on a Hadoop cluster.
4. **Pig:** Pig is an easy to understand data flow language. It helps with the analysis of large datasets which is quite the order with Hadoop. Even if one does not have the proficiency in MapReduce programming, the analysts and the persons entrusted with the task of comprehending data will still be able to analyze the data in a Hadoop cluster as the Pig scripts are automatically converted into MapReduce jobs by the Pig interpreter. MapReduce programming will be covered in detail in Chapter 8.
5. **ZooKeeper:** It is a coordination service for distributed applications.
6. **Oozie:** It is a workflow scheduler system to manage Apache Hadoop jobs.
7. ***Mahout*:** It is a scalable machine learning and data mining library.
8. **Chukwa:** It is a data collection system for managing large distributed systems.
9. **Sqoop:** It is used to transfer bulk data between Hadoop and structured data stores such as relational databases.
10. **Ambari:** It is a web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters.

4.2.5 Hadoop Distributions

Hadoop is an open-source Apache project. Anyone can freely download the core aspects of Hadoop. The core aspects of Hadoop include the following:

1. Hadoop Common
2. Hadoop Distributed File System (HDFS)
3. Hadoop YARN (Yet Another Resource Negotiator)
4. Hadoop MapReduce

There are few companies such as IBM, Amazon Web Services, Microsoft, Teradata, Hortonworks, Cloudera, etc. that have packaged Hadoop into a more easily consumable distributions or services. Although each of these companies have a slightly different strategy, the key essence remains its ability to distribute data and workloads across potentially thousands of servers thus making big data manageable data. A few Hadoop distributions are given in Figure 4.12.

4.2.6 Hadoop versus SQL

Table 4.5 lists the differences between Hadoop and SQL.

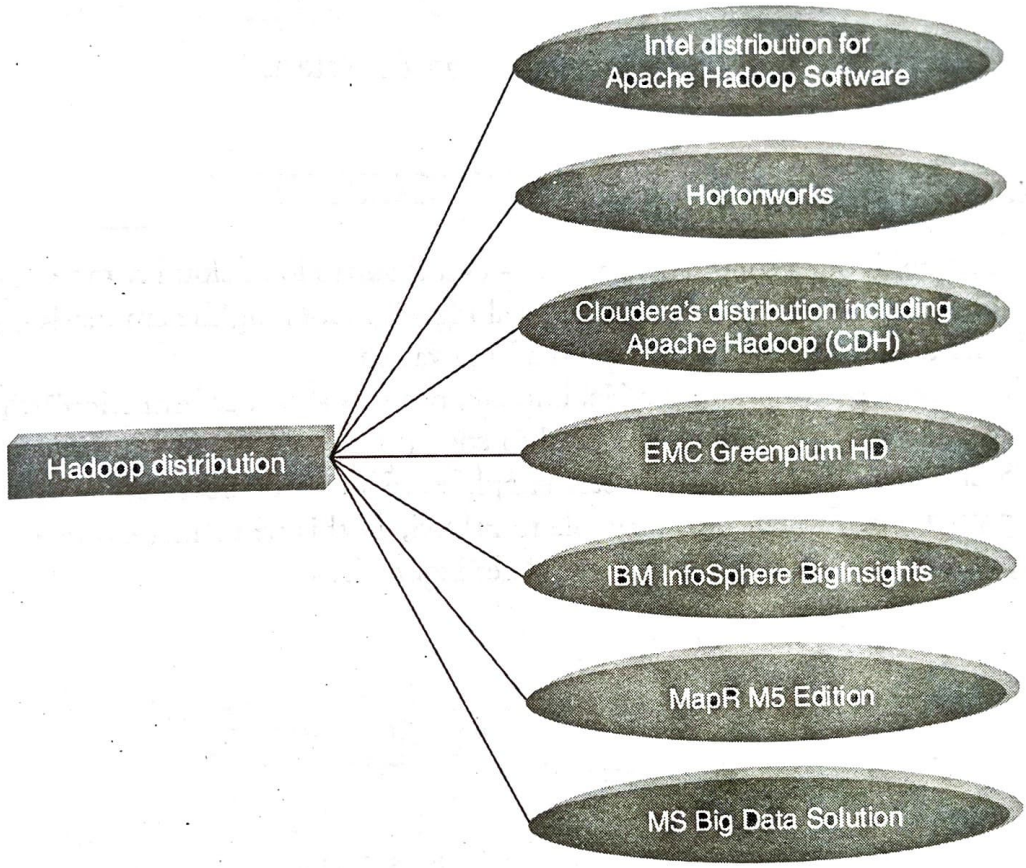


Figure 4.12 Hadoop distributions.

Table 4.5 Hadoop versus SQL

Hadoop	SQL
Scale out	Scale up
Key-Value pairs	Relational table
Functional Programming	Declarative Queries
Off-line batch processing	On-line transaction processing

4.2.7 Integrated Hadoop Systems Offered by Leading Market Vendors

Refer Figure 4.13 to get a glimpse of the leading market vendors offering integrated Hadoop systems.

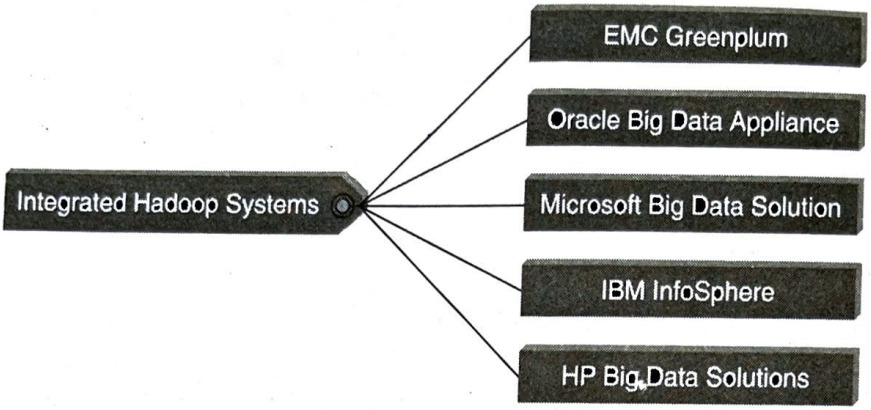


Figure 4.13 Integrated Hadoop systems.

4.2.8 Cloud-Based Hadoop Solutions

Amazon Web Services holds out a comprehensive, end-to-end portfolio of cloud computing services to help manage big data. The aim is to achieve this and more along with retaining the emphasis on reducing costs, scaling to meet demand, and accelerating the speed of innovation.

The Google Cloud Storage connector for Hadoop empowers one to perform MapReduce jobs directly on data in Google Cloud Storage, without the need to copy it to local disk and running it in the Hadoop Distributed File System (HDFS). The connector simplifies Hadoop deployment, and at the same time reduces cost and provides performance comparable to HDFS, all this while increasing reliability by eliminating the single point of failure of the name node. Refer Figure 4.14.

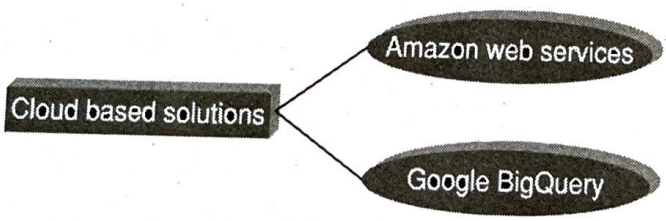


Figure 4.14 Cloud-based solutions.

REMIND ME

- NoSQL databases are non-relational, open source, distributed databases.
- NoSQL database allows insertion of data without a pre-defined schema.
- Hadoop has a shared nothing architecture.

- Hadoop 1.0 has two main parts:
  - Data storage framework
  - Data processing framework
- In Hadoop 2.0, a new and separate resource management framework called Yet Another Resource Negotiator (YARN) has been added.

POINT ME (BOOKS)

- Hadoop for Dummies, Dirk Deroos, Paul C. Zikopoulos, Roman B. Melnyk, Bruce Brown, Wiley India Pvt. Ltd.
- NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Pramod J. Sadalage and Martin Fowler.

CONNECT ME (INTERNET RESOURCES)

- http://www.mongodb.com/nosql-explained
- http://nosql-database.org/
- http://www.techrepublic.com/blog/10-things/10-things-you-should-know-about-nosql-databases/
- http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduce_Compatibility_Hadoop1_Hadoop2.html
- http://hadoop.apache.org/

TEST ME

A. Fill Me

1. The expansion for CAP is _____, _____ and _____.
2. The expansion of BASE is _____.
3. MongoDB is _____ and _____.
4. Cassandra is _____ and _____.
5. _____ has no support for ACID properties of transactions.
6. _____ is a robust database that supports ACID properties of transactions and has the scalability of NoSQL.

Answers:

1. Consistency, Availability and Partition Tolerance
2. Basically Available Soft State Eventual Consistency
3. Consistent and Partition Tolerant
4. Available and Partition Tolerant
5. NoSQL
6. NewSQL

B. Place it in the Basket

SQL	NoSQL

Following words are to be placed in the relevant basket:

(a) Relational
(b) Distributed
(c) Predefined schema
(d) Wide-column stores
(e) Vertically scalable
(f) Key–value pairs
(g) MySQL
(h) CouchDB
(i) Neo4j
(j) Cassandra
(k) Large dataset
(l) ACID properties
(m) Brewers CAP theorem
(n) Document based database
(o) Scales horizontally
(p) Avoids join operations
(q) JSON data
(r) Table or relations

Answers:

SQL	NoSQL
Relational	Distributed
Predefined schema	Wide-column stores
Vertically scalable	Key–value pairs
MySQL	CouchDB
ACID properties	Neo4j
Table or relations	Cassandra
	Large dataset
	Brewers CAP theorem
	Document based database
	Scales horizontally
	Avoids join operations
	JSON data