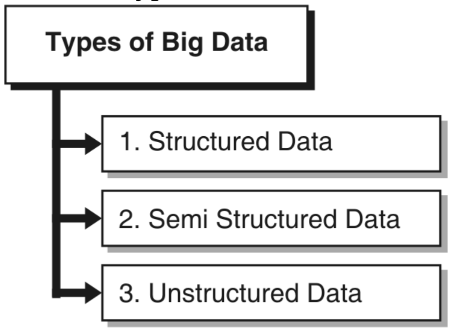
Big Data Analytics
Question Bank with Solution

1. **What is Big Data?** **(03)**

   - A massive volume of both structured and unstructured data that is so large & complex it's difficult to process with traditional database management tools & software techniques.
   - Big data simply means the huge volume of data, which is a collection of structured and unstructured data that cannot be processed using traditional computing techniques.
   - Big data is a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools.
   - The challenges include capture, cleaning, storage, search, sharing, analysis, and visualization.

2. **Explain types of Big Data.** **(04)**
   - Big data is not only a data, rather it has become a complete substance, which involves various tools, techniques and frameworks.
   - Big Data includes immense volume, high velocity, and extensible variety of data.
   - The data in it will be of three types.



- **Structured Data**
- Structured data refers to any data that resides in a fixed field within a record or file. It includes data contained in relational databases and spreadsheets.
- Example of structured data can be any kind of relational database created in Main frame, SQL server, Sybase, DB2, Oracle, Excel, Access, Terradata, Neeteza or any other.
- Example of Structured Data

Enrollment_id	Name	Mobile	DOB
100460107015	Pooja	9825033256	16/7/1995
120170107084	Roshan	9998825240	1/10/1996
141100107002	Reshma	9537417760	14/09/1998
141100107003	Tushar	9427336525	10/8/1999

➢ **Semi Structured Data**

- Semi-structured data is a form of structured data that does not conform to the formal structure of data models associated with relational databases or other forms of data tables, but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data.
- The advantages of this model is it can represent the information of some data sources that cannot be affected by schema.
- Example of semi structured data can be any XML data.

```
<?xml version='1.0'?>
<catalog>
<book id ="bk201">
<author>Rafael C. Gonzalez</author>
<title>Digital Image Processing</title>
<genre>Computer</genre>
<price>719</price>
</book>
```

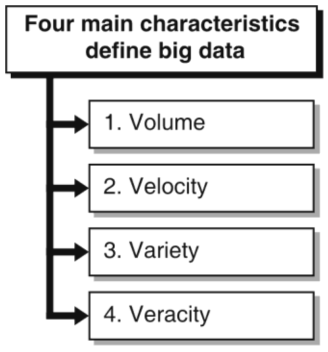
➢ **Unstructured Data**

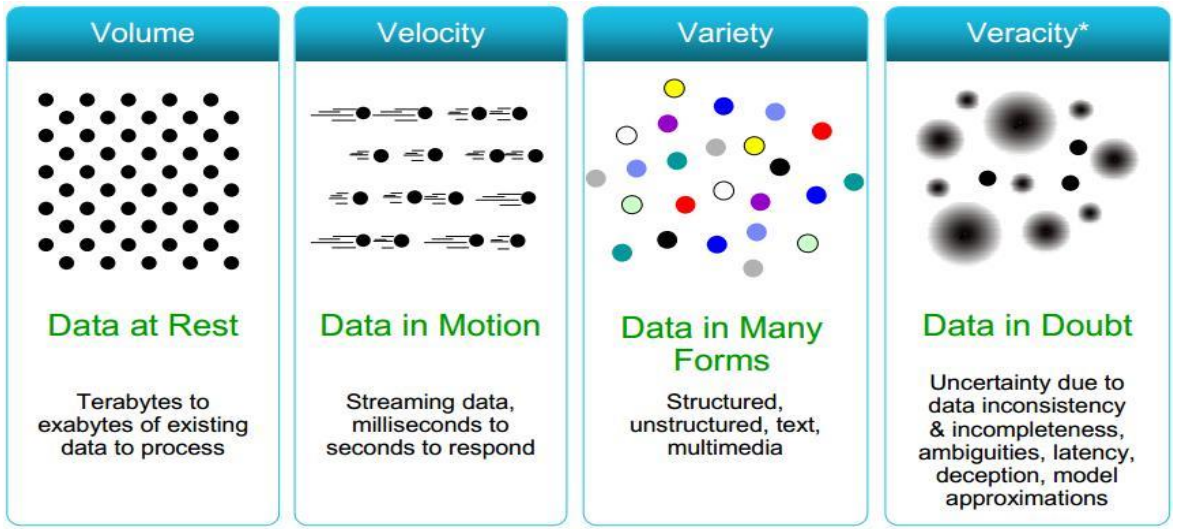
- Unstructured data refers to information that either does not have a pre-defined data model or is not organized in a pre-schema defined manner.
- Unstructured data files often include text and multimedia data. Examples include videos, photos, e-mail messages, word processing documents, audio files, presentations, web pages and many other kinds of business documents.
- Experts estimate that in any organization, 80 to 90 percent of the data is unstructured. The amount of unstructured data in enterprises is growing than structured databases.
- Some examples of human-generated unstructured data are text internal to your company, Social media data, Mobile data, website content,

```
CREATE EXTERNAL TABLE IF NOT EXISTS access_log (log_line
STRING)
PARTITIONED BY (hive_entry_timestamp STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
FIELDS TERMINATED BY '01'
STORED AS TEXTFILE
LOCATION '/usr/local/demo/access_logs';
```

3. **Explain four 'V's of Big data. <u>OR</u> Explain Characteristics of Big data.** **(04)**

The four main characteristics define big data are:




Volume
Data at Rest
Terabytes to exabytes of existing data to process
Velocity
Data in Motion
Streaming data, milliseconds to seconds to respond
Variety
Data in Many Forms
Structured, unstructured, text, multimedia
Veracity*
Data in Doubt
Uncertainty due to data inconsistency & incompleteness, ambiguities, latency, deception, model approximations

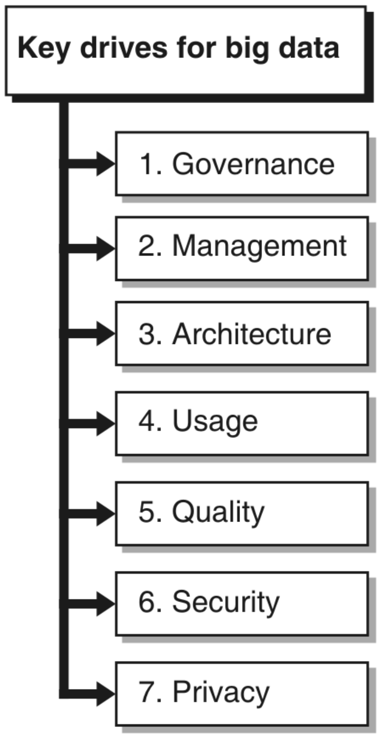
- **Volume**
  - There is exponential growth in the data storage as the data is now becoming more than only text data. Most of the data found today is in the format of videos, music and large images on social media channels. The main characteristic that makes data "big" is its volume.
  - Volume refers to the vast amounts of data that is generated every second. It is very common to have Terabytes and Petabytes of the storage system for an organization.
- **Velocity**
  - Velocity refers to the speed at which new data is generated and the speed at which it moves around. Velocity is the frequency of incoming data that needs to be processed.
  - Just think about how many likes, SMS messages, Facebook status updates, or credit card swipes are being sent on a particular telecom carrier every minute of every day, and you will have a good idea of velocity.

- A streaming application like Amazon Web Services and Kinesis and many more are example of an application that handles the velocity of data.

- **Variety**
  - Data generated from an organization are not only text data, but it includes audio file, video file, etc. This simply means that it refers to the different forms of data that we collect and use.
  - **Structured data** can be defined as a set of rules. For example, Name which can be of text type, mobile number will always be numbers; and date follow a specific pattern of date.
  - **unstructured data**, there are no rules. An image, a voice recording, a tweet or Facebook posts , they all can be of different type but express ideas and thoughts based on human understanding. One of the goals of big data is to use technology to take this unstructured data and make sense of it for better utilization.
- **Veracity**
  - Most of the company is losing a big amount in a year due to poor data management. Veracity refers to the uncertainty surrounding data, which is due to data inconsistency and incompleteness, which leads to another challenge, keeping big data organized.
  - Veracity refers to the biases, noise and abnormality in big data. It is used for data that is being stored, and mined meaningfully to the problem being analysed. In compare to volume and velocity, data analysis in Veracity is the biggest challenge.

4. Drivers for Big Data (04)

- **Key drivers behind the big data market are listed below:**



- **Governance**

Good governance focuses on consistent guidance, procedures and clear management decision-making. Organizations need to ensure standard and exhaustive data capture, they need not protect all the data, but they need to start sharing data with in-built protections with the right levels and functions of the organization.

- **Management**

Integrating and moving data across the organization is traditionally forced by data storage platforms such as relational databases or batch files with partial ability to process huge amount of data, data with difficult structure or without structure at all, or data generated or received at very high speeds.

- **Architecture**

Data architecture should be prepared in such a manner that it can break down internal storage by enabling the sharing of key data sets across the organization. It should also ensure that knowledge are being captured and relayed across to the correct set of people in the organization in a timely and accurate manner.

- **Usage**

Big data can be beneficial to a wide range of users across the organization. Executive management and boards, business operations and risk professionals, including legal, internal auditor can use the big data to analyse the facts. Not even this persons but it is also used in finance and compliance as well as customer-facing departments like sales and marketing. The key challenge is to have the ability to interpret the big amount of data that can be collated from various sources.

- **Quality**

The quality of data sets and the assumption drawn from such data sets are increasingly becoming more crucial. Organizations need to build quality and monitoring functions and parameters for big data. It is obvious that correcting a data error can be much more costly than getting the data right in the first time. Wrong data can be disastrous and much more costly to the organization if not corrected.

- **Security**

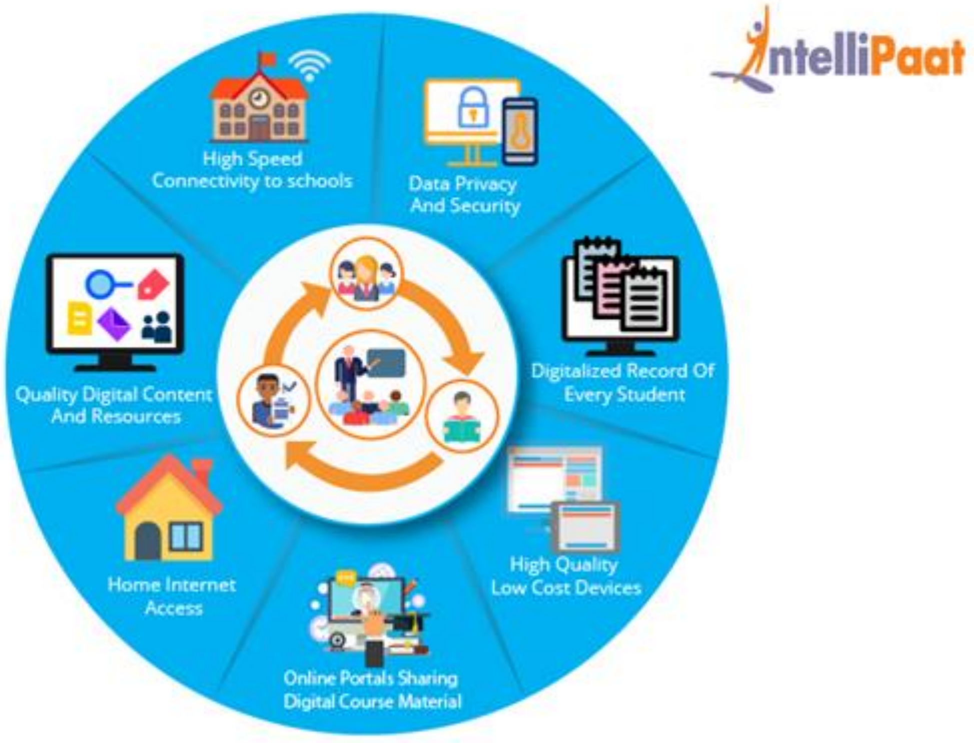
It will be definitely good for companies to start establishing security policies which are self-configurable. These policies must provide reliable relationships, and support data and resource sharing within the organizations, while ensuring that data analytics are optimized and not affected by such policies.

- **Privacy**

The increased use of big data also comes with big challenge like privacy protection. The traditional frameworks for protecting the privacy of personal information, forcing companies to audit the implementation of their privacy policies to ensure that privacy is being properly maintained in big data also.

5. Explain Applications of Big Data. (04)

- **Big Data in Education industry**


IntelliPaat
High Speed Connectivity to schools
Data Privacy And Security
Digitalized Record Of Every Student
High Quality Low Cost Devices
Online Portals Sharing Digital Course Material
Home Internet Access
Quality Digital Content And Resources

Education Industry is flooding with a huge amount of data related to students, faculties, courses, results and what not. It was not long before we realized that the proper study and analysis of this data can provide insights that can be used to improve the operational effectiveness and working of educational institutes.

Following are some of the fields in education industry that have been transformed by big data motivated changes

- **Customized and dynamic learning programs:**

Customized programs and schemes for each individual can be created using the data collected on the bases of a student's learning history to benefit all students. This improves the overall student results

- **Reframing course material:**

Reframing the course material according to the data that is collected on the basis of what student learns and to what extent by real time monitoring of what components of a course are easier to understand.

- **Grading Systems:**

New advancements in grading systems have been introduced as a result of proper analysis of student data.

- **Career prediction:**

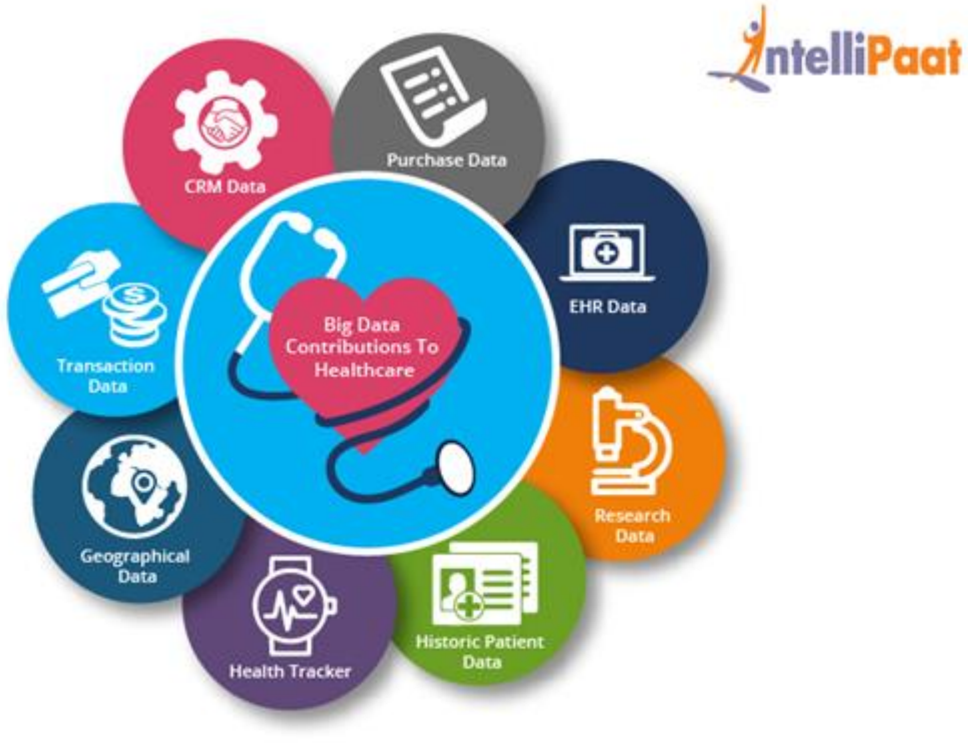
Proper analysis and study of every student's records will help in understanding the student's progress, strengths, weaknesses, interests and more. It will help in determining which career would be most appropriate for the student in the future.

The applications of big data have provided a solution to one of the biggest pitfalls in the education system, that is, the one-size-fits-all fashion of academic set up, by contributing in e-learning solutions.

Example:

The University of Alabama has more than 38000 students and an ocean of data. In the past when there were no real solutions to analyze that much data, some of that data seemed useless. Now administrators are able to use analytics and data visualizations for this data to draw out patters with students revolutionizing the university's operations, recruitment and retention efforts.

- **<u>Big data in Healthcare industry</u>**



Now healthcare is yet another industry which is bound to generate a huge amount of data. Following are some of the ways in which big data has contributed to healthcare

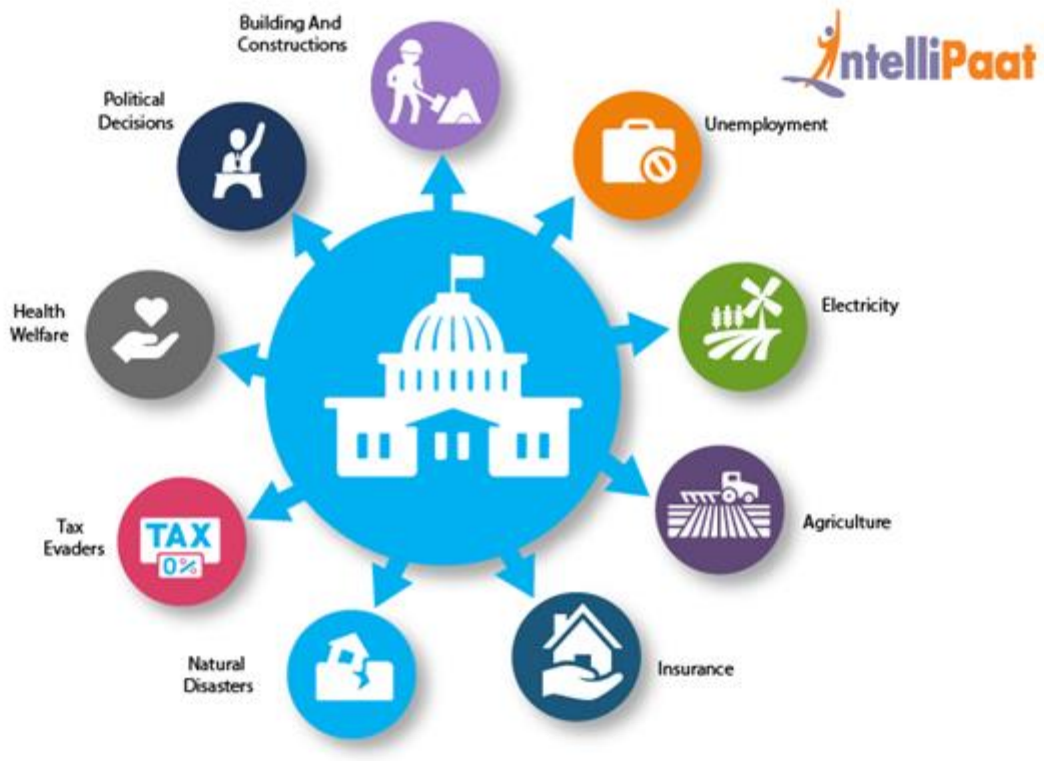
- Big data reduces costs of treatment since there is less chances of having to perform unnecessary diagnosis.
- It helps in predicting outbreaks of epidemics and also helps in deciding what preventive measures could be taken to minimize the effects of the same.
- It helps avoid preventable diseases by detecting diseases in early stages and prevents it from getting any worse which in turn makes the treatment easy and effective.
- Patients can be provided with the evidence based medicine which is identified and prescribed after doing the research of past medical results.

Example:

Wearable devices and sensors have been introduced in healthcare industry which can provide real time feed to the electronic health record of a Patient. One such technology is from Apple.

Apple has come up with what they call Apple HealthKit, CareKit and ResearchKit. The main goal is to empower the iPhone users to store and access their real time health records on their phones.

- **Big data in Government industry**



Governments, be it of any country, come face to face with a very huge amount of data on almost daily basis. Reason being, they have to keep track of various records and databases regarding the citizens, their growth, energy resources, geographical surveys and many more. All of this data contributes to big data. The proper study and analysis of this data helps the Governments in endless ways. Few of them are:

- **Welfare schemes:**

- In making faster and informed decisions regarding various political programs.
- To identify the areas that are in immediate need of attention.
- To stay up-to-date in the field of agriculture by keeping track of all the land and livestock that exists.
- To overcome national challenges such as unemployment, terrorism, energy resource exploration and more.

- **Cyber security:**
    - Big Data is hugely used for deceit recognition
    - Governments are also finding the use of big data in catching tax evaders.

Example:

The Food and Drug Administration (FDA) which runs under the jurisdiction of the Federal Government of US leverages from the analysis of big data to discover patters and associations in order to identify and examine the expected or unexpected occurrences of food based infections.

Big Data in Media and Entertainment industry

With people having access to various digital gadgets the generation of large amount of data is inevitable and this is main cause of rise in big data in media and entertainment industry.

Other than this, social media platforms are also another way in which huge amount of data is being generated. Although business in media and entertainment industry have realized the importance of this data and they have been able to leverage from it to help their businesses grow.

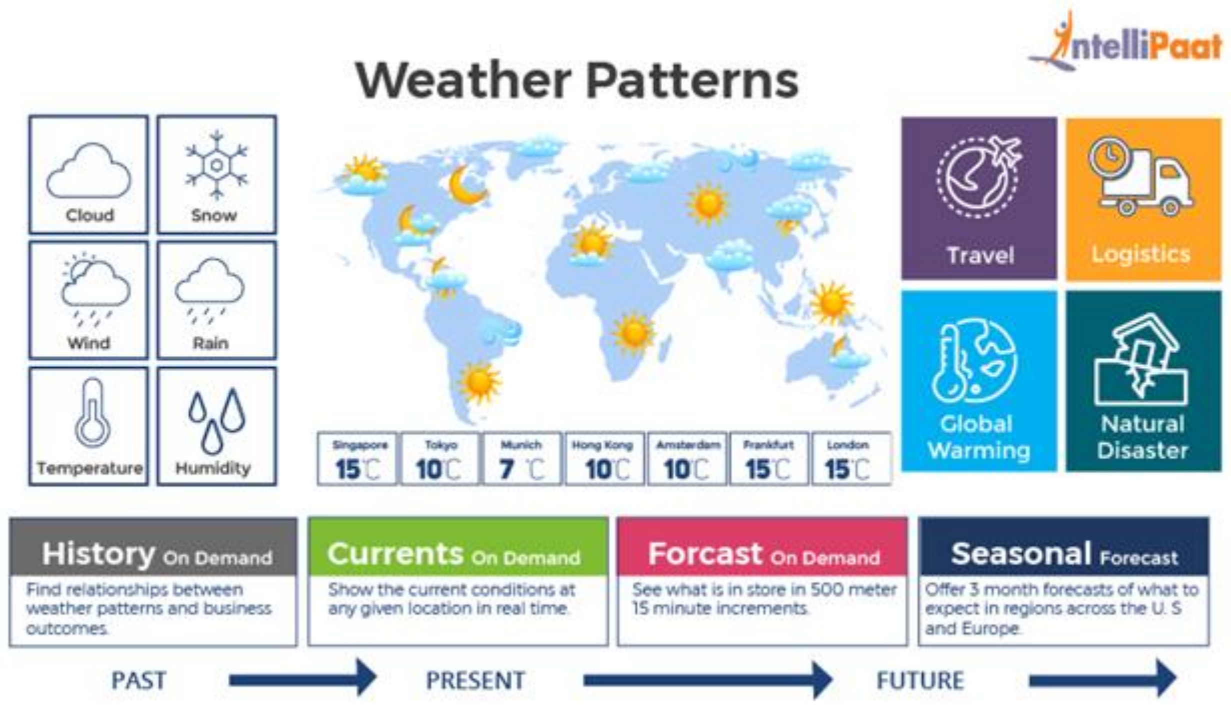
Some of the benefits extracted from the big data in media and entertainment industry:

- Predicting the interests of audiences.
- Optimized or on-demand scheduling of media streams in digital media distribution platforms.
- Getting Insights into customer’s reviews and pinpointing their animosities.
- Effective targeting of the advertisements for media

Example:

Spotify, which is an on-demand music providing platform, uses big data analytics and collects data from all of the users around the globe and then uses the analyzed data to give informed music recommendations and suggestions to every individual user. Amazon Prime, that offers, videos, music and Kindle books in a one-stop shop is also big on using big data.

Big Data in Weather patterns



There are weather sensors and satellites deployed all around the globe. A huge amount of data is collected from them and then this data is used to monitor the weather and environmental conditions.

All of the data collected from these sensors and satellites contributes to big data and can be used in different ways such as:

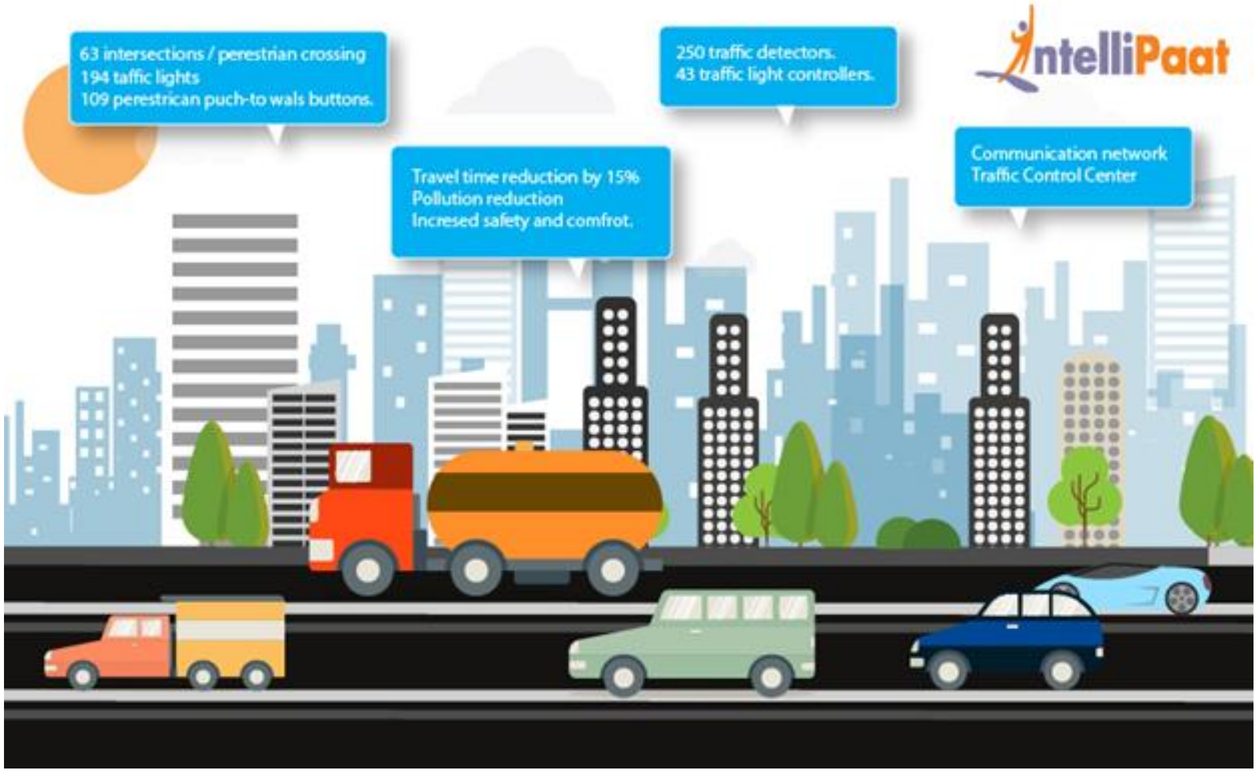
- In weather forecast
- To study global warming

- Understanding the patterns of natural disasters
- To make necessary preparations in case of crisis
- To predict the availability of usable water around the world.

Example:

IBM deep thunder which is a research project by IBM, provides weather forecasting through high performance computing of big data. IBM is also assisting Tokyo with the improved weather forecasting for natural disasters or probability of damaged power lines in order to plan successful 2020 Olympics.

Big Data in Transportation Industry


63 intersections / perestrian crossing
194 taffic lights
109 perestrican puch-to wals buttons.
250 traffic detectors.
43 traffic light controllers.
IntelliPaat
Travel time reduction by 15%
Pollution reduction
Incresed safety and comfrot.
Communication network
Traffic Control Center

Since the rise of big data, it has been used in various ways to make transportation more efficient and easy. Following are some of the areas where big data contributed to transportation.

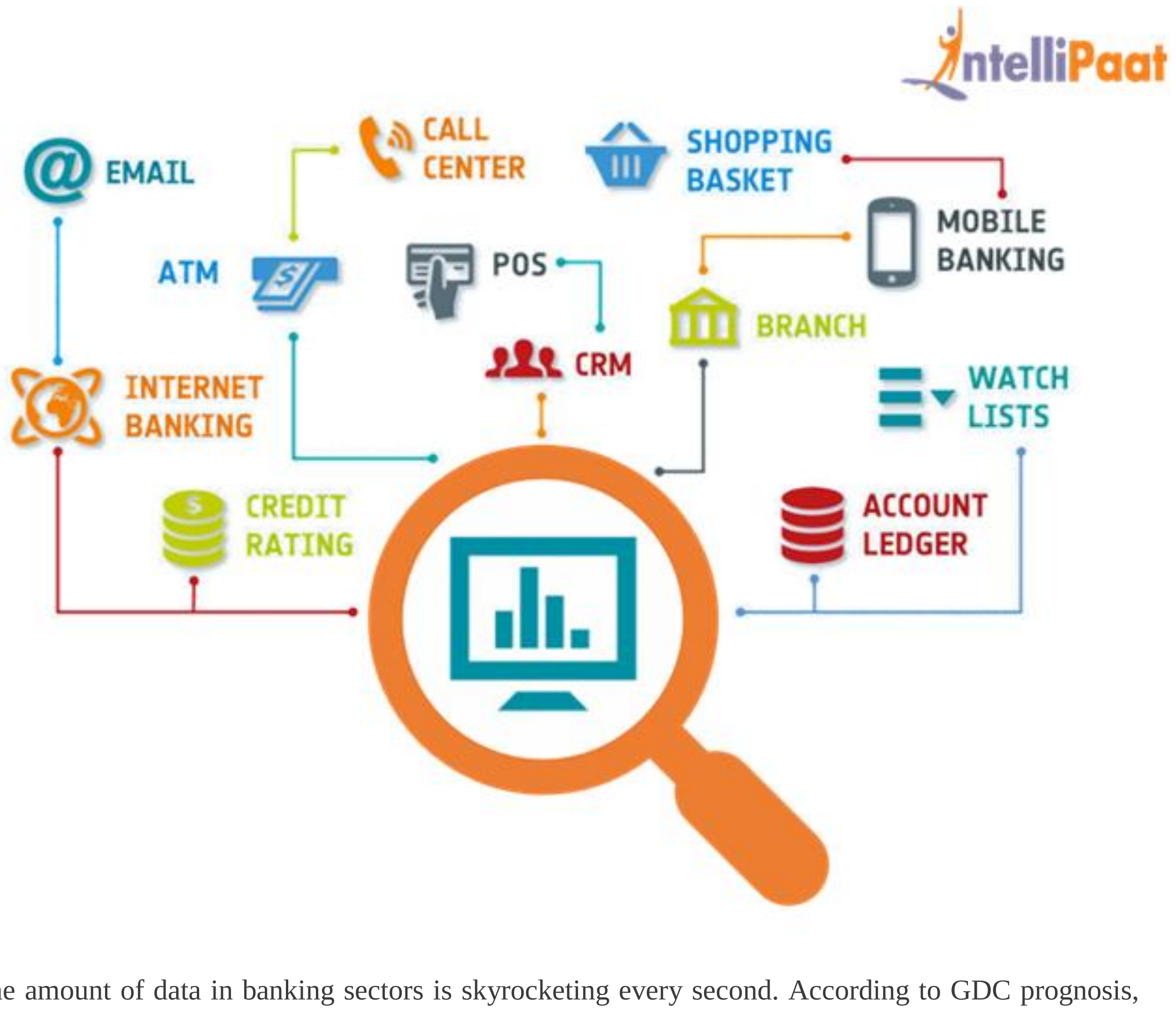
- **Route planning**: Big data can be used to understand and estimate the user's needs on different routes and on multiple modes of transportation and then utilizing route planning to reduce the users wait times.
- **Congestion management and traffic control**: Using big data, real time estimation of congestion and traffic patterns is now possible. For examples, people using Google Maps to locate the least traffic prone routes.
- **Safety level of traffic**: Using the real time processing of big data and predictive analysis to identify the traffic accidents prone areas can help reduce accidents and increase the safety level of traffic.

Example

Let’s take Uber for an example here, Uber generates and uses a huge amount of data regarding drivers, their vehicles, locations, every trip from every vehicle etc. All of this data is analyzed and then used to predict the supply, demand, location of the drivers and the fares that will be set for every trip.

And guess what? We too make use of this application when we plan route to save fuel and time, based on our knowledge of having taken that particular route sometime in the past. In this case we analyzed and made use of the data that we had previously acquired on account of our experience and then we used it to make a smart decision. It’s pretty cool that big data has played parts not only in such big fields but also in our smallest day to day life decisions too, isn’t it?

- <u>**Big Data in Banking Sector**</u>



The amount of data in banking sectors is skyrocketing every second. According to GDC prognosis, this data is estimated to grow 700% by 2020. Proper study and analysis of this data can help detect any and all the illegal activities that are being carried out such as

- The misuse of credit cards
- Misuse of debit cards

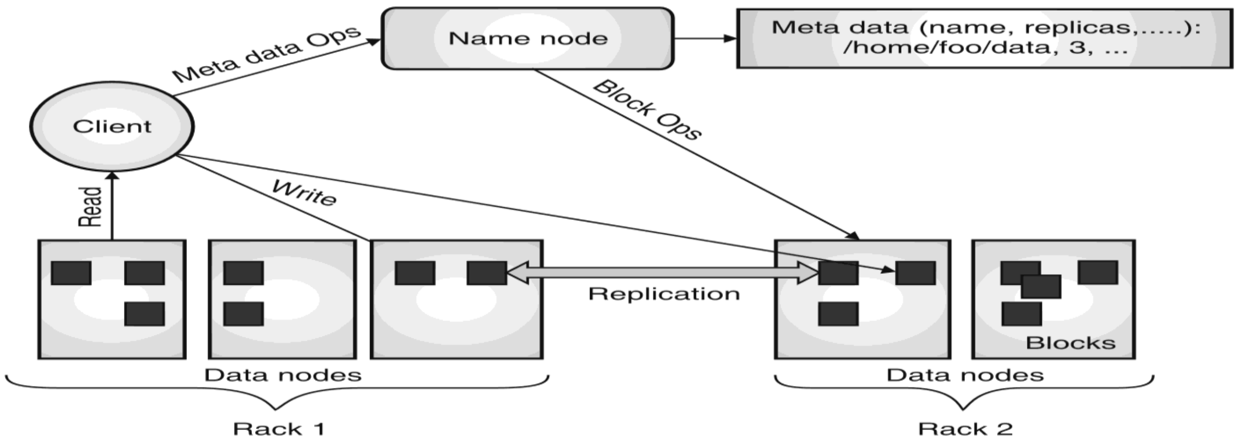
- Venture credit hazard treatment
- Business clarity
- Customer statistics alteration
- Money laundering
- Risk Mitigation

Example:

Anti-money laundering software such as SAS AML and Actimize are deployed by various financial enterprises for the main purpose of detecting suspicious transactions and analyzing customer data. One such financial enterprise is the **Bank of America** who have been a SAS AML customer for more than 25 years.

6. Explain HDFS architecture in detail. (07)

Figure shows the architecture of a Hadoop File System. HDFS follows the master-slave architecture and it has the following elements.



1. **Namenode**
   - The namenode is the main entity in HDFS. It can be also known as master. NameNode does not store the actual data or the dataset. The data itself is actually stored in the Datanodes. It stores only the metadata of HDFS that is the directory tree of all files in the file system, and tracks the file across the cluster. NameNode knows the list of the blocks and also its location for any given file in HDFS.
   - With the available information NameNode can know how to construct the file from blocks. Namenode is just like heart of HDFS, when it is down, HDFS or Hadoop cluster is inaccessible and considered down.

   Namenode can be used to perform the following task:
   - It is used to regulate client's access to files.
   - It is used to execute different file system operations such as opening, closing, and renaming files and directories.

2. **Datanode**
   - The datanode is responsible to store the actual data of HDFS. It is also known as the slave. To perform the task datanode and namenode are in constant communication. When a datanode starts, it announce itself to the namenode along with the list of blocks for which it is responsible. When a datanode is down it does not affect the availability of data or cluster because namenode will arrange for replication of the blocks which are not available by datanode.
   - Datanode is mainly configured with huge hard disk space as the actual data is stored in the datanode.

   Datanode can be used to perform the following task.
   - It is used to perform read-write operations on the file systems, as per the client request.
   - It is used to perform operations such as block creation, deletion, and replication according to the instructions received from the namenode.
3. **Block**
   - Mostly the user data is recorded in the files of HDFS. In a file system the file will be divided into one or more segments or can be stored in individual data nodes.
   - These file segments are known as blocks. In other words, Block is the minimum amount of data that HDFS can read or write. The default block size of HDFS block is 64MB, but it can be increased as per the need to change in HDFS configuration.

7. Explain Features of HDFS. (03)

1. It is suitable for the distributed storage and processing.
2. Hadoop provides a command interface to interact with HDFS.
3. The built-in servers of namenode and datanode help users to easily check the status of cluster.
4. Streaming access to file system data.
5. HDFS provides file permissions and authentication.

8. Explain Command for HDFS. (07)

The File System (FS) shell includes various shell-like commands that directly interact with the Hadoop Distributed File System (HDFS) as well as other file systems that Hadoop supports, such as Local FS, HFTP FS, S3 FS, and others. The FS shell is invoked by:

1. **AppendToFile**

   Usage : hdfs dfs -appendToFile <localsrc> ... <dst>
   Append single src, or multiple srcs from local file system to the destination file system. Also reads input from stdin and appends to destination file system.
   - hdfs dfs - appendToFile localfile /user/hadoop/hadoopfile
   - hdfs dfs - appendToFile localfile1 localfile2 /user/hadoop/hadoopfile
   - hdfs dfs - appendToFile localfile hdfs://nn.example.com/hadoop/hadoopfile
   - hdfs dfs - appendToFile - hdfs://nn.example.com /hadoop/hadoopfile Reads the input from stdin.

   Exit Code : Returns 0 on success and 1 on error.
2. **cat**

   Usage : hdfs dfs -cat URI [URI ...]
   Copies source paths to stdout.

Example

- hdfs dfs - cat hdfs://nn1.example.com/file1 hdfs: //nn2.example.com/file2
- hdfs dfs - cat file:///file3 /user/hadoop/file4

Exit Code : Returns 0 on success and -1 on error.

3. **chgrp**

Usage : hdfs dfs -chgrp [-R] GROUP URI [URI ...]
Change group association of files. The user must be the owner of files, or else a super-user.

Options

- The -R option will make the change recursively through the directory structure.

4. **chmod**

Usage : hdfs dfs -chmod [-R] <MODE[,MODE]... | OCTALMODE> URI [URI ...]
Change the permissions of files. With -R, make the change recursively through the directory structure. The user must be the owner of the file, or else a super-user.

Options

- The -R option will make the change recursively through the directory structure.

5. **chown**

Usage : hdfs dfs -chown [-R] [OWNER][:[GROUP]] URI [URI]
Change the owner of files. The user must be a super-user.

Options

- The -R option will make the change recursively through the directory structure.

6. **copyFromLocal**

Usage : hdfs dfs -copyFromLocal <localsrc> URI
Similar to put command, except that the source is restricted to a local file reference.

Options

- The -f option will overwrite the destination if it already exists.

7. **copyToLocal**

- **Usage :** hdfs dfs -copyToLocal [-ignorecrc] [-crc] URI <localdst>
- Similar to get command, except that the destination is restricted to a local file reference.

8. **count**

Usage : hdfs dfs -count [-q] <paths>

- Count the number of directories, files and bytes under the paths that match the specified file pattern. The output columns with -count are: DIR_COUNT, FILE_COUNT, CONTENT_SIZE FILE_NAME
- The output columns with -count -q are : QUOTA, REMAINING_QUATA, SPACE_QUOTA, REMAINING_SPACE_QUOTA, DIR_COUNT, FILE_COUNT, CONTENT_SIZE, FILE_NAME

Example

- hdfs dfs -count hdfs://nn1.example.com/file1 hdfs://nn2.example.com/file2
- hdfs dfs -count -q hdfs://nn1.example.com/file1

Exit Code : Returns 0 on success and -1 on error.

9. **cp**

   Usage : hdfs dfs -cp [-f] URI [URI ...] <dest>
   Copy files from source to destination. This command allows multiple sources as well in which case the destination must be a directory.

   Options
   - The -f option will overwrite the destination if it already exists.

   Example
   - hdfs dfs -cp /user/hadoop/file1 /user/hadoop/file2
   - hdfs dfs -cp /user/hadoop/file1 /user/hadoop/file2 /user/hadoop/dir

   Exit Code : Returns 0 on success and -1 on error.

10. **du**

    Usage : hdfs dfs -du [-s] [-h] URI [URI ...]
    Displays sizes of files and directories contained in the given directory or the length of a file in case its just a file.

    Options
    - The -s option will result in an aggregate summary of file lengths being displayed, rather than the individual files.
    - The -h option will format file sizes in a "human-readable" fashion (e.g 64.0m instead of 67108864)

    Example
    - hdfs dfs -du /user/hadoop/dir1 /user/hadoop/file1
    - hdfs://nn.example.com/user/hadoop/dir1

    Exit Code : Returns 0 on success and -1 on error.

11. **dus**

    Usage : hdfs dfs -dus <args>
    Displays a summary of file lengths. This is an alternate form of hdfs dfs -du -s.

12. **expunge**

    Usage : hdfs dfs -expunge
    Empty the Trash.

13. **get**

    Usage : hdfs dfs -get [-ignorecrc] [-crc] <src> <localdst>
    Copy files to the local file system. Files that fail the CRC check may be copied with the -ignorecrc option. Files and CRCs may be copied using the -crc option.

    Example
    - hdfs dfs -get /user/hadoop/file localfile
    - hdfs dfs -get hdfs://nn.example.com/user/hadoop/file localfile

    Exit Code : Returns 0 on success and -1 on error.

14. **getfacl**

    Usage : hdfs dfs -getfacl [-R] <path>

Displays the Access Control Lists (ACLs) of files and directories. If a directory has a default ACL, then getfacl also displays the default ACL.

Options

- -R : List the ACLs of all files and directories recursively.
- *path* : File or directory to list.

Examples

- hdfs dfs -getfacl /file
- hdfs dfs -getfacl -R /dir

Exit Code : Returns 0 on success and non-zero on error.

15. **getmerge**

Usage : hdfs dfs -getmerge <src> <localdst> [addnl]

Takes a source directory and a destination file as input and concatenates files in src into the destination local file. Optionally addnl can be set to enable adding a newline character at the end of each file.

16. **ls**

Usage : hdfs dfs -ls <args>

For a file returns stat on the file with the following format :

permissions number_of_replicas userid groupid filesize modification_date modification_time filename

For a directory it returns list of its direct children as in Unix. A directory is listed as :

permissions userid groupid modification_date modification_time dirname

Example

hdfs dfs -ls /user/hadoop/file1

Exit Code : Returns 0 on success and -1 on error.

17. **lsr**

Usage : hdfs dfs -lsr <args>

Recursive version of ls. Similar to Unix ls -R.

18. **mkdir**

Usage : hdfs dfs -mkdir [-p] <paths>

Takes path uri's as argument and creates directories.

Options

The -p option behavior is much like Unix mkdir -p, creating parent directories along the path.

Example

- hdfs dfs -mkdir /user/hadoop/dir1 /user/hadoop/dir2
- hdfs dfs -mkdir dfs://nn1.example.com/user/hadoop/dir
- hdfs://nn2.example.com/user/hadoop/dir

Exit Code : Returns 0 on success and -1 on error.

19. **moveFromLocal**

Usage : dfs -moveFromLocal <localsrc> <dst>

Similar to put command, except that the source localsrc is deleted after it's copied.

20. **moveToLocal**

Usage : hdfs dfs -moveToLocal [-crc] <src> <dst>
Displays a "Not implemented yet" message.

21. mv

Usage : hdfs dfs -mv URI [URI ...] <dest>
Moves files from source to destination. This command allows multiple sources as well in which case the destination needs to be a directory. Moving files across file systems is not permitted.

Example

- hdfs dfs -mv /user/hadoop/file1 /user/hadoop/file2
- hdfs dfs -mv hdfs://nn.example.com/file1 hdfs://nn.example.com/file2 hdfs://nn.example.com/file3 hdfs://nn.example.com/dir1

Exit Code : Returns 0 on success and -1 on error.

22. put

Usage : hdfs dfs -put <localsrc> ... <dst>
Copy single src, or multiple srcs from local file system to the destination file system. Also reads input from stdin and writes to destination file system.

- hdfs dfs -put localfile /user/hadoop/hadoopfile
- hdfs dfs -put localfile1 localfile2 /user /hadoop /hadoopdir
- hdfs dfs -put localfile hdfs://nn.example.com /hadoop /hadoopfile
- hdfs dfs -put - hdfs://nn.example.com /hadoop /hadoopfile Reads the input from stdin.

Exit Code : Returns 0 on success and -1 on error.

23. rm

Usage : hdfs dfs -rm [-skipTrash] URI [URI ...]
Delete files specified as args. Only deletes non empty directory and files. If the -skipTrash option is specified, the trash, if enabled, will be bypassed and the specified file(s) deleted immediately. This can be useful when it is necessary to delete files from an over-quota directory. Refer to rmr for recursive deletes.

Example

hdfs dfs -rm hdfs://nn.example.com/file /user /hadoop /emptydir

Exit Code : Returns 0 on success and -1 on error.

24. rmr

Usage : hdfs dfs -rmr [-skipTrash] URI [URI ...]
Recursive version of delete. If the -skipTrash option is specified, the trash, if enabled, will be bypassed and the specified file(s) deleted immediately. This can be useful when it is necessary to delete files from an over-quota directory.

Example

- hdfs dfs -rmr /user/hadoop/dir
- hdfs dfs -rmr hdfs://nn.example.com/user/hadoop/dir

Exit Code : Returns 0 on success and -1 on error.

25. setfacl

Usage : hdfs dfs -setfacl [-R] [-b|-k -m|-x <acl_spec> <path>]|[--set <acl_spec> <path>]

Sets Access Control Lists (ACLs) of files and directories.

Options

- -b : Remove all but the base ACL entries. The entries for user, group and others are retained for compatibility with permission bits.
- -k : Remove the default ACL.
- -R : Apply operations to all files and directories recursively.
- -m : Modify ACL. New entries are added to the ACL, and existing entries are retained.
- -x : Remove specified ACL entries. Other ACL entries are retained.
- --set : Fully replace the ACL, discarding all existing entries. The *acl_spec* must include entries for user, group, and others for compatibility with permission bits.
- *acl_spec* : Comma separated list of ACL entries.
- *Path* : File or directory to modify.

Examples

- hdfs dfs -setfacl -m user:hadoop:rw- /file
- hdfs dfs -setfacl -x user:hadoop /file
- hdfs dfs -setfacl -b /file
- hdfs dfs -setfacl -k /dir
- hdfs dfs -setfacl --set user::rw-,user:hadoop:rw-,group::r--,other::r-- /file
- hdfs dfs -setfacl -R -m user:hadoop:r-x /dir
- hdfs dfs -setfacl -m default:user:hadoop:r-x /dir

Exit Code : Returns 0 on success and non-zero on error.

26. **setrep**

Usage : hdfs dfs -setrep [-R] [-w] <numReplicas> <path>

Changes the replication factor of a file. If *path* is a directory then the command recursively changes the replication factor of all files under the directory tree rooted at *path*.

Options

- The -w flag requests that the command wait for the replication to complete. This can potentially take a very long time.
- The -R flag is accepted for backwards compatibility. It has no effect.

Example

hdfs dfs -setrep -w 3 /user/hadoop/dir1

Exit Code : Returns 0 on success and -1 on error.

27. **stat**

Usage : hdfs dfs -stat URI [URI ...]

Returns the stat information on the path.

Example

hdfs dfs -stat path

Exit Code : Returns 0 on success and -1 on error.

28. **tail**

Usage : hdfs dfs -tail [-f] URI

Displays last kilobyte of the file to stdout.

Options

The -f option will output appended data as the file grows, as in Unix.

Example

hdfs dfs -tail pathname

Exit Code : Returns 0 on success and -1 on error.

29. test

Usage : hdfs dfs -test -[ezd] URI

Options

- The -e option will check to see if the file exists, returning 0 if true.
- The -z option will check to see if the file is zero length, returning 0 if true.
- The -d option will check to see if the path is directory, returning 0 if true.

Example :

Hdfs dfs -test -e filename

30. text

Usage : hdfs dfs -text <src>

Takes a source file and outputs the file in text format. The allowed formats are zip and TextRecordInputStream.

31. touchz

Usage : hdfs dfs -touchz URI [URI ...]

Create a file of zero length.

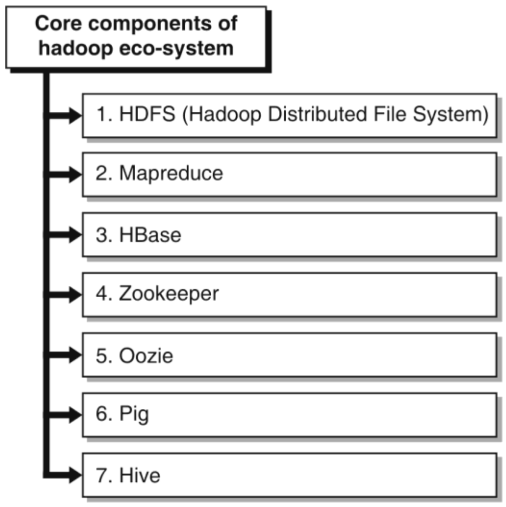
Example :

hadoop - touchz pathname

Exit Code : Returns 0 on success and -1 on error.

9. What are the advantages of Hadoop? Explain Hadoop architecture and its component with proper diagram.

- Hadoop's Writable-based serialization is able to reduce the object-creation overhead by reusing the Writable objects, which is not possible with the Java's native serialization framework.


Core components of hadoop eco-system
1. HDFS (Hadoop Distributed File System)
2. Mapreduce
3. HBase
4. Zookeeper
5. Oozie
6. Pig
7. Hive

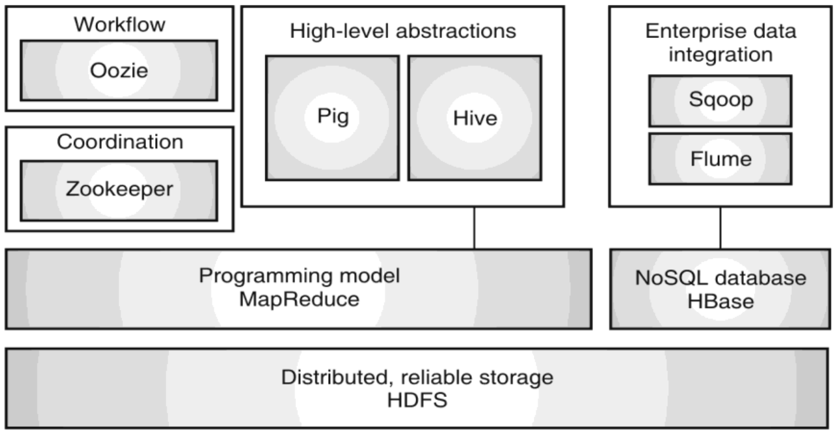
Hadoop ecosystem consists of different components. The core components of Hadoop ecosystem is shown in Figure.

1. **HDFS (Hadoop Distributed File System)**

   A fundamental component of the Hadoop ecosystem is the Hadoop Distributed File System (HDFS). HDFS is the mechanism by which a large amount of data can be distributed over a cluster of computers, and data is written once, but read many times for analytics. It provides the foundation for other tools, such as HBase

2. **MapReduce**

   Hadoop's main execution framework is MapReduce, a programming model for distributed, parallel data processing, breaking jobs into mapping phases and reduce phases (thus the name). Developers write MapReduce jobs for Hadoop, using data stored in HDFS for fast data access. Because of the nature of how MapReduce works, Hadoop brings the processing to the data in a parallel fashion, resulting in fast implementation.



3. **HBase**

   A column-oriented NoSQL database built on top of HDFS, HBase is used for fast read/write access to large amounts of data. HBase uses Zookeeper for its management to ensure that all of its components are up and running.

4. **Zookeeper**

   Zookeeper is Hadoop's distributed coordination service. Designed to run over a cluster of machines, it is a highly available service used for the management of Hadoop operations, and many components of Hadoop depend on it.

5. **Oozie**

   A scalable workflow system, Oozie is integrated into the Hadoop stack, and is used to coordinate execution of multiple MapReduce jobs. It is capable of managing a significant amount of complexity, basing execution on external events that include timing and presence of required data.

6. **Pig**

   An abstraction over the complexity of MapReduce programming, the Pig platform includes an execution environment and a scripting language (Pig Latin) used to analyze Hadoop data sets. Its compiler translates Pig Latin into sequences of MapReduce programs.

7. **Hive**
   An SQL-like, high-level language used to run queries on data stored in Hadoop, Hive enables developers not familiar with MapReduce to write data queries that are translated into MapReduce jobs in Hadoop. Like Pig, Hive was developed as an abstraction layer, but geared more toward database analysts more familiar with SQL than Java programming.

The Hadoop ecosystem also contains several frameworks for integration with the rest of the enterprise:

1. **Sqoop**
   It is a connectivity tool for moving data between relational databases and data warehouses and Hadoop. Sqoop leverages database to describe the schema for the imported/ exported data and MapReduce for parallelization operation and fault tolerance.
2. **Flume**
   It is a distributed, reliable, and highly available service for efficiently collecting, aggregating, and moving large amounts of data from individual machines to HDFS. It is based on a simple and flexible architecture, and provides a streaming of data flows. It leverages a simple extensible data model, allowing you to move data from multiple machines within an enterprise into Hadoop.

Beyond the core components shown in Figure. **Hadoop's ecosystem is growing to provide newer capabilities and components, such as the following:**

1. **Whirr**
   This is a set of libraries that allows users to easily spin-up Hadoop clusters on top of Amazon EC2, Rackspace, or any virtual infrastructure.
2. **Mahout**
   This is a machine-learning and data-mining library that provides MapReduce implementations for popular algorithms used for clustering, regression testing, and statistical modeling.
3. **BigTop**
   This is a formal process and framework for packaging and interoperability testing of Hadoop's sub-projects and related components.
4. **Ambari**
   This is a project aimed at simplifying Hadoop management by providing support for provisioning, managing, and monitoring Hadoop clusters.

10. What is Map Reduce ? What are its Advantages ? (04)

Hadoop MapReduce is a software framework for easily writing applications which process huge amounts of data in-parallel on large clusters of commodity hardware in a reliable and fault-tolerant manner.

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

1. The **Map task** takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).
2. The **Reduce task** takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples. The reduce task is always performed after the map job.

Advantages of Map Reduce

MapReduce is useful for batch processing on terabytes or peta bytes of data stored in Apache Hadoop.

The following table describes some of the MapReduce key benefits:

Benefit	Description
Simplicity	Developers have freedom to write applications in their language of choice, such as Java, C++ or Python, and MapReduce jobs are easy to run.
Scalability	MapReduce can process huge amount of data (in terms of petabytes and more) , stored in HDFS on one cluster
Speed	It uses parallel processing it means that MapReduce can take problems that used to take days to solve which can be solved in hours or minutes
Recovery	MapReduce also takes care of failures. If a machine with one copy of the data is unavailable than another machine has a copy of the same key/value pair, which can be used to solve the same sub-task. The JobTracker keeps track of these all processes.
Minimal data motion	MapReduce moves are able to compute processes to the data on HDFS and not the other way around. Processing tasks can occur on the physical node where the data resides. This significantly reduces the network I/O patterns and contributes to Hadoop's processing speed.

11. What is NoSQL database? List the differences between NoSQL and relational databases. Explain in brief various types of NoSQL databases in practice. (07)

➢ **NoSQL Database**

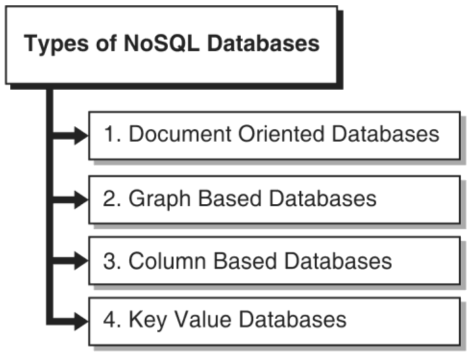
- NoSQL stands for "Not Only SQL"
- A NoSQL (originally referring to "non SQL", "non relational" or "not only SQL") is an approach for data management and database design that is useful for very large sets of distributed data.
- **Definition of NoSQL:** NoSQL databases are highly scalable and flexible database management systems which allow you to store and process unstructured as well as semi-structured data which is not possible through RDBMS tools.
- Some people think this term opposes SQL, but that is not the case. NoSQL systems are non-relational and can coexist with relational databases. They are suitable in applications where a large amount of data is involved. Data in this case is either structured, unstructured or semi-structured.
- NoSQL is particularly useful for storing unstructured data, which is growing far more rapidly than structured data and does not fit the relational schemas of RDBMS.
- Common types of unstructured data include: user and session data; chat, messaging, and log data; time series data such as IOT and device data; and large objects such as video and images.

Differences between NoSQL and relational databases:

Criteria	Relational Database Management	NoSQL Database Management
Data model	Tables and schemas	Partition Keys to retrieve data
ACID properties	Strictly followed	No strict adherence
Scalability	Vertical scalability	Horizontal scalability
Data manipulation	Using queries in SQL and executed by RDBMS	Using object-based APIs
Velocity of data	Moderate	Very high
Suitability	Structured data	Structured, semi-structured and unstructured data

Types of NoSQL Databases:

There have been various approaches to classify NoSQL databases. The classification of NoSQL databases is done on the base of data model used in it. NoSQL databases typically fall into following categories:



➢ **Document Oriented Databases**

- Document oriented databases treat a document as a whole and avoid separating a document in its fundamental name/value pairs. At a collection level, this allows for putting together a discrete set of documents into a single collection.
- Document databases allow indexing of documents on the basis of not only its primary identifier but also its properties. Many open-source document databases are available today in the market but the most noticeable among the available options are MongoDB and CouchDB. As a matter of fact, MongoDB has become one of the most popular NoSQL databases.

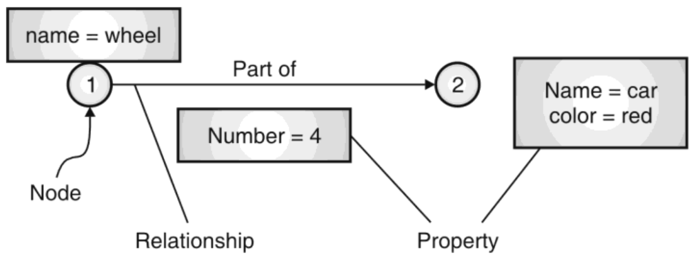
Structure of Document Oriented Database

```
{officeName:" Cygnet Infotech Pvt. Ltd.”,
{Street: “6-Swastik Society, City:"Ahmedabad”, State:"Gujarat", Pincode:" 380009 "}
}
{officeName: " Cygnet Infotech Pvt Ltd.  ",
{Boulevard:" C-101/102-10, C Wing 1st Floor”, Block:" Pune IT Park ", City: “Pune”,
Pincode: 411020”}
}
{officeName:" Cygnet Infotech Pvt Ltd ",
{Latitude:" 72.5644° E ", Longitude:" 23.0388° N "}
}
```

- **Example :** Apache CouchDB, Clusterpoint, Couchbase, DocumentDB, HyperDex, IBM Domino, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB

➢ **Graph Based Databases**

- A graph database uses graph structures with nodes, edges, and properties to represent and store data. By definition, a graph database is any storage system that provides index-free adjacency. This means that every element contains a direct pointer to its adjoining element and no index lookups are necessary.
- General graph databases that can store any graph are distinct from specialized graph databases such as triple-stores and network databases. Indexes are used for traversing the graph.



- The above mentioned sample shows that is the symbol for nodes and properties are shown in the rectangular boxes. In above given example, node 1 and node 2 is related with each other. node 2 is in relationship with node 1. Wheel is the part of car. Any car will have four wheels. so, in above example you can see number = 4 as a property of object car.
- **Example :** AllegroGraph, ArangoDB, Neo4j, GraphDB, OrientDB, Stardog, etc.

➢ **Column Based Databases**

- The column-oriented storage allows data to be stored effectively. It avoids consuming space when storing nulls by simply not storing a column when a value does not exist for that column.
- Each unit of data can be thought of as a set of key/value pairs, where the unit itself is identified with the help of a primary identifier, often referred to as the primary key. Big table and its clones tend to call this primary key as the row-key
- Big Table, for instance is a high performance, compressed and proprietary data storage system owned by Google.

It has the following attributes :

- **Sparse :** Some cells can be empty.
- **Distributed :** Data is partitioned across many hosts.
- **Persistent :** Stored to disk.
- **Multidimensional :** More than 1 dimension.
- **Map :** Key and value.
- **Sorted :** Maps are generally not sorted but this one is.

– A 2-dimensional table comprising of rows and columns is part of the relational database system.

City	Pincode	Strength	Project
Ahmedabad	380009	250	20
Pune	411020	200	15

– For above RDBMS table a BigTable map can be visualized as shown below.

Structure

```
Cygnet_Ambd{
city: Ahmedabad
pincode: 380009
},
details: {
strength: 250
projects: 20
}
Cygnet_Pune: {
address: {
city: Pune
pincode: 411020},
details: {
strength: 200
projects: 15
}
},
```

– The outermost keys Ahmedabd and Pune are correspondent to rows.

– ‘address’ and ‘details’ are called **column families**.

– The column-family ‘address’ has **columns** ‘city’ and ‘pincode’.

– The column-family details’ has **columns** ‘strength’ and ‘projects’.

– Columns can be referenced using column Family.

– **Example :** Google’s BigTable, HBase and Cassandra are the most popular column store based databases.

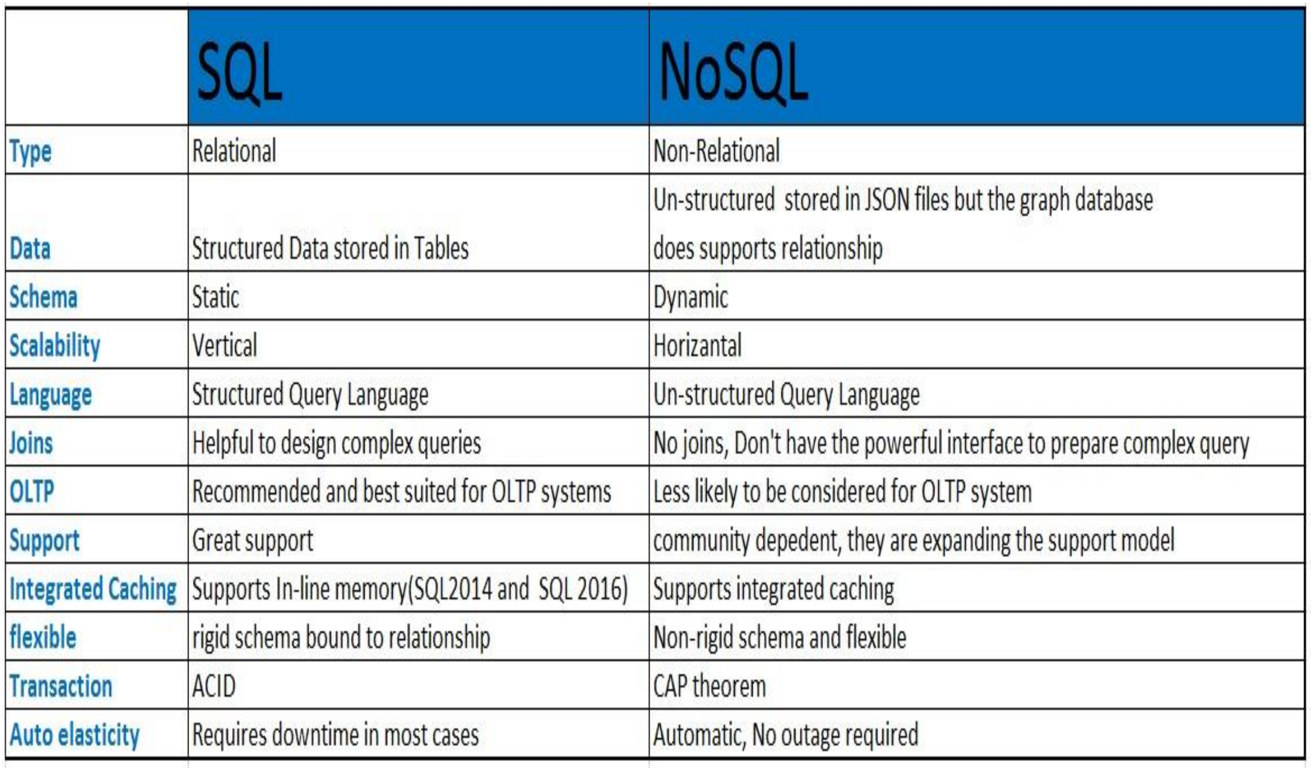
- **Key Value Databases**

- The key of a key/value pair is a unique value in the set and can be easily looked up to access the data. Key/value pairs are of diverse types: some keep the data in memory and some provide the capability to prevail the data to disk. A simple, yet powerful, key/value store is Oracle's Berkeley DB.
- **Structure :** Consider the data subset represented in the following table. Here the key is the name of the Cygnet InfoTech's country name, while the value is a list of addresses of Cygnet InfoTech centers in that country.

Key	Value
"India"	{" 16-Swastik Society, Nr. AMCO Bank, Stadium Circle, Navrangpura, Ahmedabad 380009, Gujarat, India"}
"USA"	{" Mack-Cali Centre III, 140 E, Ridgewood Avenue, Suite 415 ST, Paramus, NJ 07652"}
"Australia"	{" Suite 301, Level 3, 171 Clarence Street, Sydney, NSW 2000"}

- This key/value type database allow clients to read and write values using a key as follows :
  - Get(key), returns the value associated with the provided key.
  - Put(key, value), associates the value with the key.
  - Multi-get(key1, key2, .., keyN), returns the list of values associated with the list of keys.
  - Delete(key), removes the entry for the key from the data store.
- Key/value type database has some pros and cons. One of its disadvantage is that the model will not provide any kind of traditional database capabilities such as ACID properties like atomicity of transactions, or consistency when multiple transactions are executed simultaneously. Such capabilities must be provided by the application itself.
- Secondly, as the size of data increases, it is possible that maintaining unique values as keys may become more difficult; To overcome this issue requires the introduction of some complexity in generating character strings that will remain unique among an extremely large set of keys.
- **Example :** Membase, Redis, MemcacheDB, etc.

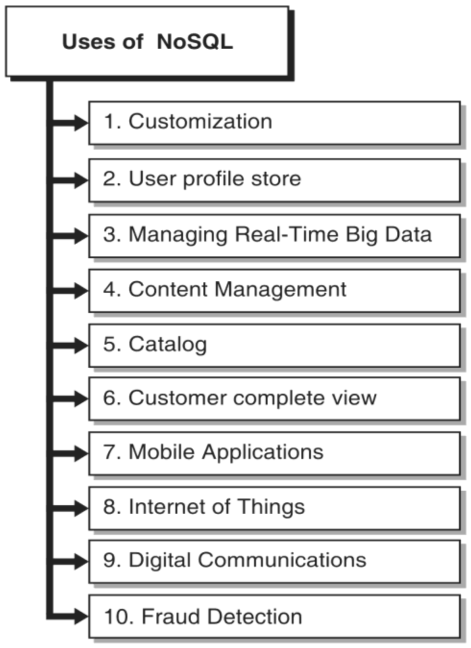
12. Write differences between NoSQL and SQL. (04)



	SQL	NoSQL
Type	Relational	Non-Relational
Data	Structured Data stored in Tables	Un-structured stored in JSON files but the graph database does supports relationship
Schema	Static	Dynamic
Scalability	Vertical	Horizantal
Language	Structured Query Language	Un-structured Query Language
Joins	Helpful to design complex queries	No joins, Don't have the powerful interface to prepare complex query
OLTP	Recommended and best suited for OLTP systems	Less likely to be considered for OLTP system
Support	Great support	community depedent, they are expanding the support model
Integrated Caching	Supports In-line memory(SQL2014 and SQL 2016)	Supports integrated caching
flexible	rigid schema bound to relationship	Non-rigid schema and flexible
Transaction	ACID	CAP theorem
Auto elasticity	Requires downtime in most cases	Automatic, No outage required

13. Use of NoSQL in industry (07)

NoSQL was evolved to meet the lacking requirement of traditional RDBMS system. The key features of NoSQL like flexibility, scalability, agile structure and better performance make it most suitable for current industries to work upon. The big problem of today's industry is big data. NoSQL is the best suitable solution to manage these type of big and multidimensional data. Following are the uses for which industry uses NoSQL.



1. **Customization**
   Managing session information using traditional RDBMS has been a pain point for many web application developers, especially as applications have grown in scale. In those cases, a global session store which manages session information for individual user who visits the website. NoSQL has emerged as one of the best options for storing web app session information because of key value storing properties of it. The unstructured nature of session data is easier to store in a schema-less document of NoSQL rather than storing it in a structured and more rigid RDBMS record. Session information can be used to provide customized solution to user.
2. **User profile store**
   User profile management is important to web and mobile applications to enable online transactions, user preferences, user authentication and more. Today, web and mobile applications support millions or even hundreds of millions of users. While relational databases are inefficient in serving big amount of user data as they are limited to a single server while distributed databases can scale out across multiple servers. With NoSQL, capacity is increased simply by adding commodity servers, making it far manageable and less expensive to scale.
3. **Managing Real-Time Big Data**
   The ability to extract information from operational data in real-time is critical for any enterprise. It increases operational efficiency, reduces costs, and increases revenue by enabling user to act immediately on current data. In the past, operational databases and analytical databases were maintained as different environments. Today, NoSQL is used as both the front-end for storing and managing operational data from any source, and then output is used for Hadoop , as well as back end for receiving, storing and serving analytical results from Hadoop.
4. **Content Management**
   The key to effective content is the ability to select a variety of content, combine it and present it to the customer at the moment of interaction. NoSQL document databases, with their dynamic and flexible data model, are perfect for storing any type of content like structured, semi-structured or unstructured data. NoSQL document databases do not require any schema to be defined first. It enable enterprises to easily create and produce new types of content, it

also enables them to assimilate user-generated content, such as comments, images, or videos posted on social media, with the same ease and agility.

5. **Catalog**
   Catalogs are referenced by web and mobile applications both. Each enterprise offer more products and services, even collect more reference data, so catalogs become fragmented by application and business unit. As relational databases rely on fixed schema architecture, it's not uncommon for multiple applications to access multiple databases, which introduces complexity and data management challenges. NoSQL document database because of its high agility, enables enterprises to more easily aggregate catalog data within a single database.
6. **Customer complete view**
   Users always expect a consistent experience regardless of technology, while the enterprise wants to focus on sell opportunities and to provide the highest level of customer service. As the number of products and services, technology, brands and business units increases, the fixed schema architecture of relational databases forces enterprises to fragment user data because diverse applications work with diverse customer data. NoSQL document databases provide dynamic data model to enable multiple applications to access the same customer data and can add new attributes without affecting other applications.
7. **Mobile Applications**
   With nearly two billion smart phone users face scalability challenges in terms of growth and volume. For instance, it is usual for mobile games to reach tens of millions of users in very short time. With a distributed, scale-out database, mobile applications can start with a small deployment and expand as the user base grows, instead of deploying an expensive, large relational database server from the beginning.
8. **Internet of Things**
   The volume, velocity, and variety of machine-generated data are increasing with the generation of digital telemetry, which is semi-structured and continuous. Relational databases struggle with the three well-known challenges from big data IoT applications: scalability, throughput, and data variety. By contrast, NoSQL allows enterprises to scale concurrent data access to millions of connected devices and systems, store big volumes of data, and meet the performance requirements of target critical infrastructure and operations.
9. **Digital Communications**
   In an enterprise environment, digital conversation may take the form of online interaction via direct messaging to help visitors find a product or complete the checkout process. Relational databases are limited in responsiveness and scalability while NoSQL databases, because of their distributed architecture, deliver the sub-millisecond responsiveness and scalability that digital communication applications require.
10. **Fraud Detection**
    For financial service organizations, fraud detection is crucial for reducing economical loss and complying with regulations. Immediate confirmation is expected by customers when they pay with a credit or debit card. This process impacts both the enterprise and its customers. Relational databases lack to meet this requirement while NoSQL databases can reliably deliver the required performance.