ISSN Print: 2394-7500
ISSN Online: 2394-5869
Impact Factor: 8.4
IJAR 2023; 9(4): 26-33
www.allresearchjournal.com
Received: 22-01-2023
Accepted: 25-02-2023

Dr. T Muthukumar
Professor (Business Analytics) & Associate Dean (Academic)
Xavier Institute of Management & Entrepreneurship (XIME), Bangalore, Karnataka, India

Corresponding Author:
Dr. T Muthukumar
Professor (Business Analytics) & Associate Dean (Academic)
Xavier Institute of Management & Entrepreneurship (XIME), Bangalore, Karnataka, India

Integrating E-Governance with big data analytics using apache spark

Dr. T Muthukumar

Abstract
The constant innovations and rapid developments in the IT industry have revolutionized the thinking and mindset of the people throughout the world. Government departments have also been computerized to provide transparent, efficient and responsible government through e-governance. The government have been providing access to various websites or portal for filing complaints, uploading or downloading forms, pictures, data or PDFs to avail the government services. Enlightened citizens are frequently using the portal to access government services. Thus, the size and volume of data that need to be managed by government departments have been increasing drastically under e-governance. The traditional database management system is not designed to deal with such mix type of data. Moreover, the speed at which the e-governance generated data need to be processed is another big challenge being faced by traditional database system.
All the above said concerns can be solved by using the emerging technology - Big Data Analytics techniques. Big data analytic techniques can make the government more efficient and transparent by processing structured, unstructured or mixed types data at a great speed. In this paper, we shall understand the scenario for the need or the emergence of big data analytics in e- governance and knowhow of Apache Spark. This paper proposes a practical approach to integrate big data analytics with e- governance using Apache Spark. This paper also reflects how major issues of traditional database management system (mixed type datasets, speed and accuracy) can be resolved through the integration of big data analytics and e-governance.

Keywords: E-governance, big data analytics, apache spark

1. Introduction
Due to the advancement in technology, various industries or domains like transport, tourism, hotel, banks and so on have been digitized and generating large amount of data. People are using the Internet to generate forms, reports, graphs, periodic or to do online shopping on discounted rates. Social media (Facebook, Instagram, blogs, twitter etc.) or entertainment industries are using computers to share pictures, audios and videos. According to a general survey posted on Wikipedia till April 2019, 56.1% of population has been accessing Internet services [10]. Government websites have also been generating massive amount of data by uploading or downloading pictures or credentials like finger prints, retina scan, forms, reports of the citizens. The big data analytic techniques have been designed to store, process and analyze such a mixed mode data.
Thus, integrating e-governance with big data is the need of the hour. The main objective of this research paper is to provide an insight of emerging needs to deal with huge data under e-governance, introduction to Apache Spark framework to store, process and deal with big data at greater speed and accuracy. This paper progresses by expanding the literature review on e-governance, Big data, Big data analytics, followed by Apache Spark and proposed a new system framework.

2. Literature Review
As both the terms – e-governance and Big data analytics are very vast, let us try to understand them one by one. The first section will deal with e-governance and second section deal with big data.

A. E-governance

E-governance refers to the process of providing government services online. It makes the government system more efficient, transparent and accountable. Citizens can access government services by using the web portals that have been created to provide all services at one click. They can easily upload or download forms, photos, data and so on. The biggest example of the success of e-governance is Aadhar (UDAI) portal run by Indian government. This portal stores variety of data like text, images, PDF, retina scan, name, age, address and other related information of their stakeholders [2]. As of 2018, India has a population of over 1.355 billion people, and its growth is expected to continue through at least 2050 [11]. Managing such mix type of data being generated at such a huge level is the biggest challenge.

B. Big data

As the name is indicating, a huge amount of data that is very difficult to store, analyze and execute is called Big data. The definition of big data varies from company to company. One company's big data can be small for others. But when the data does not fit in memory, nor on hard disk and if there is continuous demand for processing, then it is considered as big data [5]. For some companies, data up to 10 TB is considered as big data. While, for some other companies, 1 PB of data is considered as big data [3, 12].

i. Big data characteristics

Fig. 1 shows the basic characteristics of big data. These characteristics are also known as 3Vs [4, 12].

1. Volume: It refers to the total size of data set.
2. Variety: It refers to the type of data generated these days.
3. Velocity: It refers to the speed and frequency at which data gets generated.

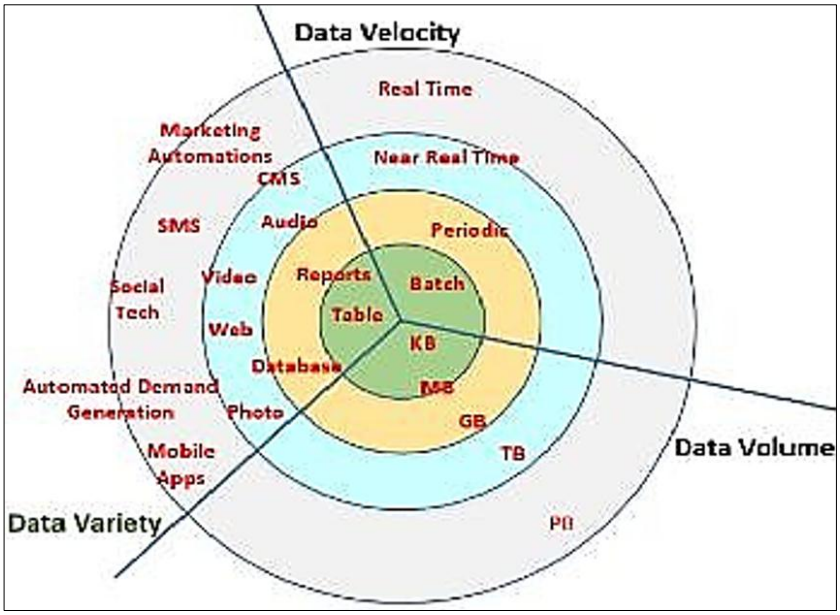

Fig 1: Big data characteristics

Veracity has also been considered as 4th characteristics (4Vs) of big data. It refers to the uncertainty and trust worthiness of data's origin [1].

ii. Types of big data

The big data can be in the form of text, audio, video, blog, PDF, log files, sensor data etc. Thus, the data sets can be categorized [8] as under:

(1) Structured: This type of data has fixed patterns, formats or schema which can be managed using RDBMS (traditional database management system).
(2) Semi-structured: This type of database does not have pre-defined patterns or formats. It may contain data represented through graphs.
(3) Unstructured: This type of data does not have any standards or formats of data. Different variety of data like text, images, video, audio, PDf, log files etc. come under this category. Here we hardly find any direct relationship between these datasets.

C. Big data analytic

The term Big data analytics refers to the process of analyzing raw datasets to understand their hidden patterns and behaviors using qualitative and quantitative techniques. The analytic techniques are basically used in B2C (Business to Customer) applications to collect, categorize, store, process and analyze the trends and future expectations. Thus, Big data also helps in decision making. The various techniques of Big data analytics are as follows in Fig. 2:

1. Descriptive analytics: It uses historic or traditional datasets and apply predictive or trends analysis on them.
2. Predictive analysis: It indicates what will happen in future. It also indicates what will be the situation, trends or outcome in that particular time span.
3. Prescriptive analysis: It helps to take best possible solution from multiple options. Predictive analytics become much mature and stable with age and experience.

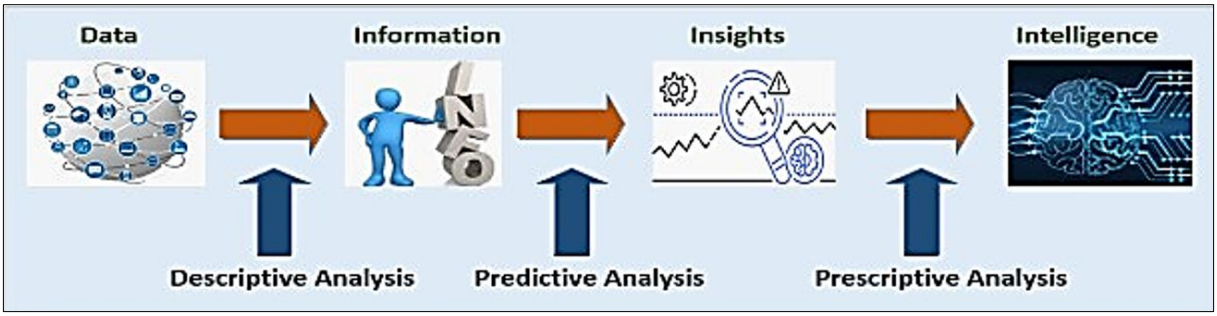

Fig 2: Big data analytic techniques

The Big data analytics uses any of the above said techniques to find hidden patterns and correlation. The traditional data analytic tools like UNIX, R become inefficient as they store and analyze the data on a single machine. A single machine can never be able to store and process the big data. There is multiple open source software available to process the data stored on multiple computers. One such solution is HDFS (Hadoop Distributed File System) and Hadoop Map Reduce [6, 7]. The HDFS helps to store the data on multiple computers called clusters. Clustering makes the data more secure and fault-tolerant [9]. MapReduce is used to perform parallel processing of big data through clustering [13]. Hadoop distributed File System and MapReduce provide excellent results in storing, processing and analyzing big data. But they face challenges when the matter comes of the processing speed and scale of big data. Another technique is Apache Spark that provides faster processing at lesser time than MapReduce.

3. Apache spark

Apache Spark is an open-source, cluster-computing framework that provides in-memory processing of large amount of data. As compare to MapReduce, Apache Spark's processing capabilities are almost 100 times faster due to in-memory computation and 10 times faster while using the disk for storing input-output or processing [12, 13]. Apache MapReduce always stores or keep primary, intermediate data on hard disks. Thus, to-and-fro travelling of data consumes time and makes processing speed slower than Apache Spark.

Apache Spark has powerful APIs that help to correlate the unstructured, structured and semi-structured data, to analyze and evaluate the data to make future predictions. Thus, it is a helpful tool in decision-making. It is highly scalable, having scalability up to 200 PB of storage. HDFS is also highly reliable and fault tolerant system. A single cluster of HDFS, containing 4500 servers, can support billion files and blocks [1].

A. Apache Spark ecosystem

Just like HDFS, Apache Spark does not provide any storage or resource management capabilities. Still it manages to process the data at a great speed and less time that seems almost real-time processing. Let us understand its framework and ecosystem to know more about it as shown in Fig. 3. Apache Spark framework is divided into three layer [14] such as:

- Spark Core
- Spark Ecosystem
- Resource Management

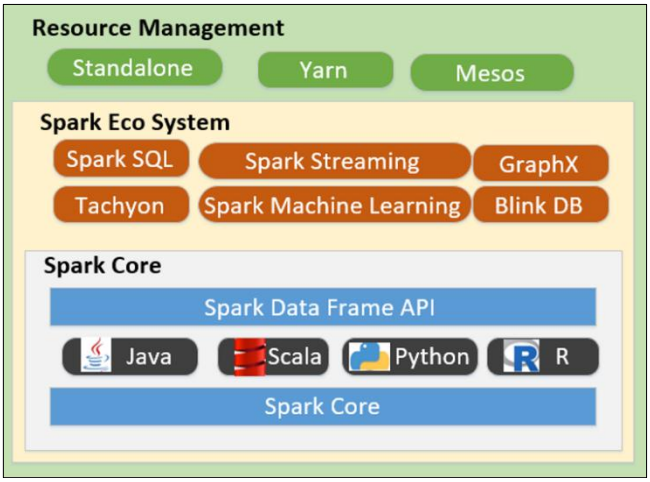

Fig 3: Apache Spark Ecosystem

i. Spark Core layer

Spark Core is the foundation layer of the framework [15]. This layer is responsible for basic input-output tasks, scheduling, etc. Apache Spark is written using an Object Oriented Language – Scala that runs on the top of JVM (Java Virtual Machine). It supports various programming languages like Java, Python and R programming. Spark has a very powerful query engine that responses the query data in real time manner. Spark provides Data Frame APIs in Scala, Java and Python languages to use this query engine.

ii. Spark Ecosystem layer

Spark Ecosystem provides another additional library on the top of Spark Core and Spark Data Frame APIs to enhance query processing in real-time. Here the data is divided and distributed into the form of partitions on various clusters. This is also known as RDD (Resilient Distributed Datasets). The various components of Spark Ecosystem are as follows:

- **Spark SQL:** It provides SQL like interface to query data in CSV, JSON etc. formats. It helps to execute Data Frames using JDBC APIs and also executes structured or unstructured type of data.
- **Spark Streaming:** It refers to the capability of taking and executing data in micro batches (micro RDD) to provide real-time execution [16].
- **Blink DB:** It is just like another query engine. It is used to execute the interactive queries from large datasets at faster speed but with the tendency of having errors. This query engine is useful where aggregated values may contain errors or be less accurate.
- **Spark Machine Learning(MLib):** It is Spark's machine learning libraries that contains common learning algorithms like culturing, classification, collaborative filtering, regression etc.

- **Graph:** Apache Spark provides distributed graph processing API on the top of Spark Core. It provides various components like subgraph, Join Vertices and optimized variant of Pregl API and graph algorithms to simplify graph analytics
- **Tachyon**: It refers to process of storing distributed file system on cache memory. It stores all the required files/data/intermediate results in cache memory. This helps to provide file sharing and faster execution in less time.

iii. Resource management

Before understanding the Apache Spark resource management, it is good to understand Hadoop Distributed File System (HDFS) and MapReduce.

The Hadoop Distributed File System (HDFS) is a Java based file system. It has been designed to store large amount of data. Here the data is stored on multiple computers, called commodity hardware, in redundant manner to provide high fault tolerance at cheaper cost. The data of the file is split into three equal sizes of 128 MB each [1]. Each part is stored on different commodity hardware, so that the data can be recovered in case of failure of one hardware. This framework is based upon Master-Slave architecture, consisted of Name Node and Data Node. The Name Node acts as a master. It is responsible for mapping of data, tracking of data, opening, closing or calling the data files. Data Node, as a slave, is used to store and manage the nodes.

Apache Spark uses Hadoop for storing big data files. The different methods of using Apache Spark with Hadoop are as shown in Fig. 4.

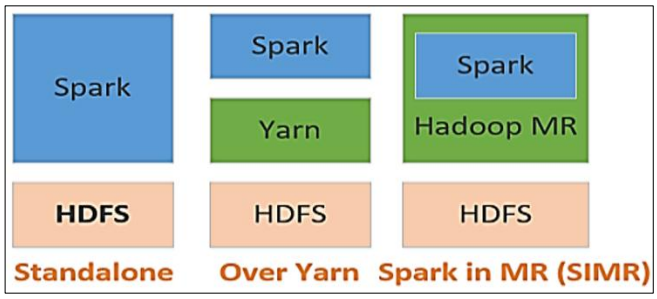

Fig 4: Different methods of using Spark

- **Standalone**: In standalone method, HDFS is stored separately and Spark is stored at the top of it. Spark and HDFS work side by side to perform all the required tasks.
- **Hadoop Yarn**: In this type of deployment, Spark works or access HDFS with the help of Yarn for resource management. It does not need to be pre-installed or any administrative rights. This way Spark and HDFS can be easily integrated to take maximum benefits using Yarn.
- **Spark Map Reduce**: In this type of deployment, a user can start Spark and takes its advantages without installing them. Spark in Mad Reduce helps to use Spark API's without administrative rights.

Apache Spark combines the features of both HDFS and Map Reduce. Just like Apache Hadoop, Apache Spark does not provide any resource management like YARN. It manages the resources with a single node cluster setup for less complex datasets but need to be integrated with resource management modules to deal with complex, distributed cluster datasets as shown using Fig. 5(a) and 5(b):

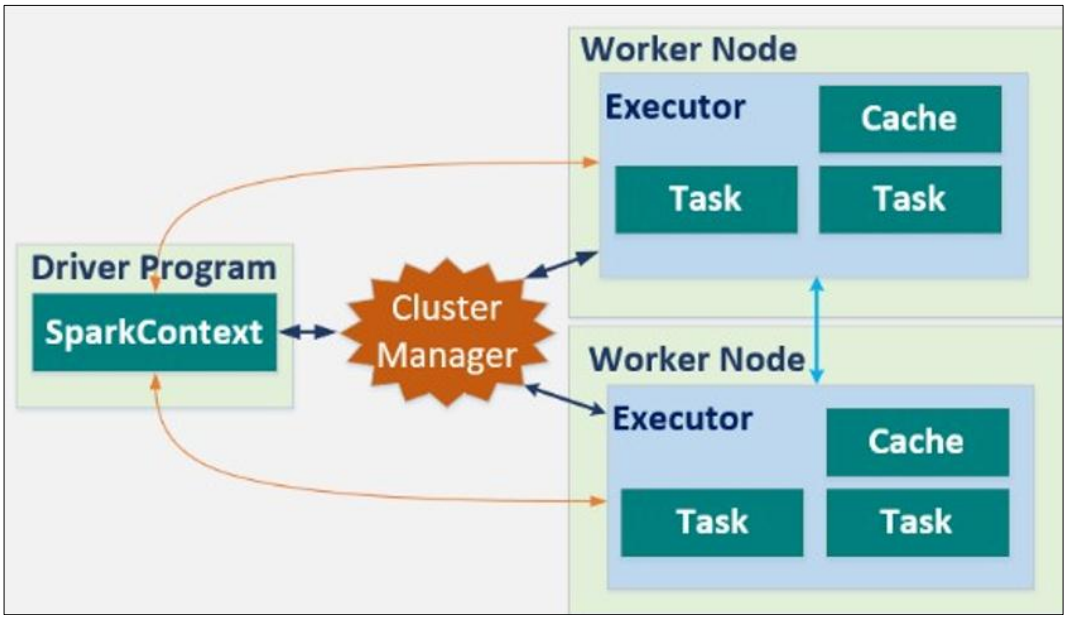

Fig 5(a): Apache Spark cluster architecture

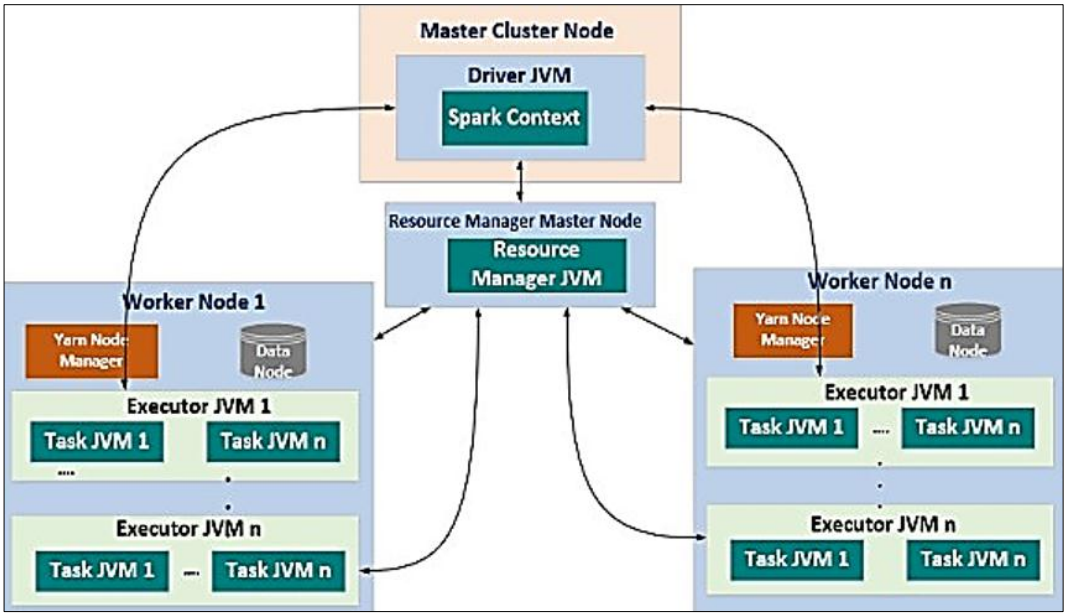

Fig 5(b): Apache Spark cluster architecture

B. Apache Spark cluster architecture

The important points related to the execution of Apache Spark cluster architecture [14, 17] are as follows.

(1) Apache Spark Cluster is based on master-salve architecture with two main processes.

(2) Master Daemon: The master or the main node contains the actual program (driver program) that drives the application. The first task done by this driver program is to create a Spark Context object that is similar to creating a database connection. This Spark Context helps to start and continue the communication between two ends.

The Spark Context helps to divide the jobs (RDDs) into tasks. These tasks are further distributed and stored on the worker nodes.

Worker Daemon: The worker node is the slave node whose main responsibilities are to receive, execute the

(4) source manager starts searching for the node where the required data resides. It splits the job into different stages and each stage into different tasks.

(5) Master daemon supplies all the requisite details to the slave node(s) before execution. Worker daemon keeps the constant watch on the execution and sends the status updates to the Master daemon simultaneously.

(6) The same steps are followed for all the slave nodes. Once all the requirements are completed then all the slave nodes provide the consolidated value to the Master Daemon.

Proposed Apache spark based system for analyzing e-governance datasets

The e-governance can be integrated with various components provided by Hadoop to face the challenges occurred due to big data. Here, a new system is proposed to design and integrate Apache Spark with Hadoop to provide a framework to deal e-governance big data at a greater speed and accuracy.

A. Experimental environment set up

The basic requisites of the proposed system are:

1) **Download and install Java**: In order to install Apache Spark on the system, you need to download and install Java (version 1.8). Use a secured link from the Internet to download Java.

2) **Download and install Scala**: Apache Spark is made up of Scala language. Use the secured link to download Scala. To install Scala [18], run the following commands (see Table I) for extracting the tar files:

Table I: Installing Scala

Sr. No	Requirement	Command(s)
1	Extracting Scala tar file	$ tar xvf scala- <version>.tgz Example: $ tar xvf scala- 2.11.6.tgz
2	Moving Scala to a specific directory	# cd /home/file/downloads/ # mv scala- 2.11.6/usr/local/scala # exit
3	Setting path for Scala	$ export PATH =$PATH:/usr/local/scala/bin
4	Verifying Scala installation	$ scala -version

RDDs and return the output back to the Spark Context. It may reside on the Spark master node or on separate worker nodes. Basically, it runs on the particular node where the required data is actually stored to take the advantage of data locality. All the processing takes places on the local JVM.

Resource manager like Yarn or Mesas may reside on the master node itself or on a separate machine which works between Master and Slave nodes. They help the Spark Contact object to manage to-and-fro movement of data and instructions. The resource manager is responsible for the tracking of in-between processes.
Spark master daemon initiates the processing by creating a Spark Context object and sends the request to the resource manager (Yarn/Mesas) first. The Spark Context object exists till the end of the program.

Download and install Apache Spark: Use the secured link to download the latest version of Apache Spark. To install Apache Spark, run these commands (Table II).

Table 2: Installing Apache Spark

Sr. No	Requirement	Command
1	Extracting Apache Spark tar file	$ tar xvf spark-<version>.tgz Example: $ tar xvf spark-1.3.1-bin-hadoop2.6.tgz
2	Moving Spark to a specific directory	# cd /home/hadoop/downloads/ # mv spark-1.3.1-bin-hadoop2.6/usr/local/spark # exit
3	Setting path for Spark	$ export PATH =$PATH:/usr/local/spark/bin
4	Verifying Spark installation	$ spark -shell

B. Data source
The data source, used in this paper, has been taken from open-source data repository (data. World) for research and analysis. This government collected dataset has been published by National University of Educational Planning and Administration, on behalf of department under Ministry of Human Resource Development.

Department of School Education andLiteracy, Government of India.
The current dataset is regarding the status of Elementary Education in India, published in the year 2014-2015 and 2015-2016. The School Report Cards are available at www.schoolreportcards.in(https://data.world/inderz/india-district-level-school-report-card). Use Table III to know the detail of the dataset files, their sizes and number of records.

Table 3: Dataset description

File Name	Size of File	Number of Columns	Number of Records
Dist. Report Card 2014-15	2 MB	256	680
Dist. Report Card 2015-16	2 MB	256	680

Some of the fields from the selected dataset, for both the years, are shown in Table IV.

Table 4: Fields from School Report Card

AC_YEAR	Data Reported From	Total Schools by Category	Total Schools by Category - Government & Aided	Schools by Category: Boys Only
Schools by Category: Girls Only	Enrolment by School Category	Teachers by School Category	Single-Classroom Schools by School Category	Schools Approachable by All Weather Road
Schools with Computer	Single Teacher Schools	Enrolment by Grade	Teachers by School Category: Male	Girls Enrolment By School Category
Teachers by School Category: Female	Number of Classrooms by School Category	Committee (Government & Aided Schools)	Schools with Enrolment <= 50	Schools Constituted School Management

Table 5: Records from dataset

Ac_YEAR	Date reported from			School with computer (2014-15)							
				Primary only	Primary with	Primary with	Upper Primary	Upper Primary	Primary with	Upper Primary	Total
	DISTCD	State Name	Dist Name	SCOMO 1	SCOMO 2	SCOMO 3	SCOMO 4	SCOMO 5	SCOMO 6	SCOMO 7	SCOMO TOT
2014-15	0101	Jammu and Kashmir	Kupwara	21	91	1	1	1	38	14	167
2014-15	0102	Jammu and Kashmir	Baramula	16	116	6	0	3	76	24	241
2014-15	0103	Jammu and Kashmir	Srinagar	26	193	45	0	2	204	2	472
2014-15	0104	Jammu and Kashmir	Badgam	15	94	8	3	1	67	15	203
2014-15	0105	Jammu and Kashmir	Pulwama	16	89	4	1	2	64	13	189
2014-15	0106	Jammu and Kashmir	Anantnag	52	160	11	1	3	85	13	325
2014-15	0107	Jammu and Kashmir	Leh (Ladakh)	19	84	2	2	2	35	1	145
2014-15	0108	Jammu and Kashmir	Kargil	18	65	6	2	1	19	15	126
2014-15	0109	Jammu and Kashmir	Doda	13	35	9	0	4	38	2	101
2014-15	0110	Jammu and Kashmir	Udhampur	23	87	30	0	4	63	0	207

Analysis is done to find out which State of India has highest percentage increase in Computers in schools. The fields that are required to be studied for this analysis is AC_YEAR, Data Reported from and Schools with Computer as shown in Table V.

C. Experiment details

The purpose of this study is to understand the increase or decrease of number of computers in each and every state of India for the last 2 years. Then try to find the state whose percentage of using computer has been increased.

i. Algorithm

On the basis of available datasets following generic algorithm is designed to read, load and analysis the datasets.

Step 1: Convert the downloaded Excel files into the flat files (.csv).

Step 2: Load the *.csv* file in to the system to create RDD; to create load Data Frames.

Step 3: Run the SQL commands on Data Framesto perform manipulations.

Step 4 Rum the SQL command to extract the subset data file containing fields - Year, Data Reported from and Schools with Computer 2014-15.

Step 5 Repeat the Step 1 to Step 4 to extract the similar dataset for Year 2015-16.

Step 6 Load the aggregated subset files for both the years using Spark framework.

Step 7 Iterate the files for all the districts of each state using Select command and find total number of computers available in each state for both years.

Step 8 Iterate state-wise to compare the Total Computers in each state.

Step 9 Calculate percentage increase or decrease in the total number of computers in schools for each state as

Formula for percentage change in computer per state = [(Total Computer in 2015-16 per state - Total Computer in 2014-15 per state) / (Total Computer in 2014-15 per state)] * 100

ii. Implementation and result

The implementation and the corresponding results of the experimental environment are as follows:

- Files are loaded in Spark based experimental environment having 1 Executor, 1 Node system with Windows 10, Intel Core i5, 2.4 GHz, 8 GB RAM.
- Project Setup
- Here major dependencies are on the following factors:
  - Spark core
  - Spark sq.
  - Spark csv
- Querying *.csv* data is very easy using the Spark *csv* library. For that, we will be using SQL Context object. With SQL Context object, we can query the data like we do in any database language. We can perform all the operations on data like SELECT and also write the data into a new file.
- After setting up an SBT project, we will start by adding required dependencies into built. sbt.
- Project execution

To execute the project, perform the following tasks

(1) Set up the Apache Spark configuration.
(2) Initiate the process by making the spark Context object.
(3) Make the sql Context object to retrieve the desired dataset from the *.csv* files.
(4) Read the *.csv* files using sql Context .read .format() method. Load the data from *.csv* file into a Resilient Distributed Dataset (RDD).

Create the Data Frames objects using method toDF();

(5) Run the appropriate queries using SQL commands to retrieve the desired dataset.
(6) Place the ResultSet in the temporary table or can save it to use it later.

- **Execution Time**
- After implementing the Algorithm steps, the following is the result displayed by the Spark-shell: Result is [state : BIHAR, old_SCOMPTOT = 5296, new SCOMPTOT =6085, difference 789, % diff = 14.89803625377]
- Overall experiment completed in 2-4 seconds with Spark-shell on a single node
- Around 6-8 seconds, when program is compiled to make jar and then executed on more than one node

iii. Hypothetical analysis

The current dataset has been analyzed on a Windows based Spark-shell. Considering the amount of data to be processed, single node system has been used. However, if data volume increases in size, the datasets can be distributed to N nodes (multiple nodes) and the overall processing will be as shown in Fig. 6.

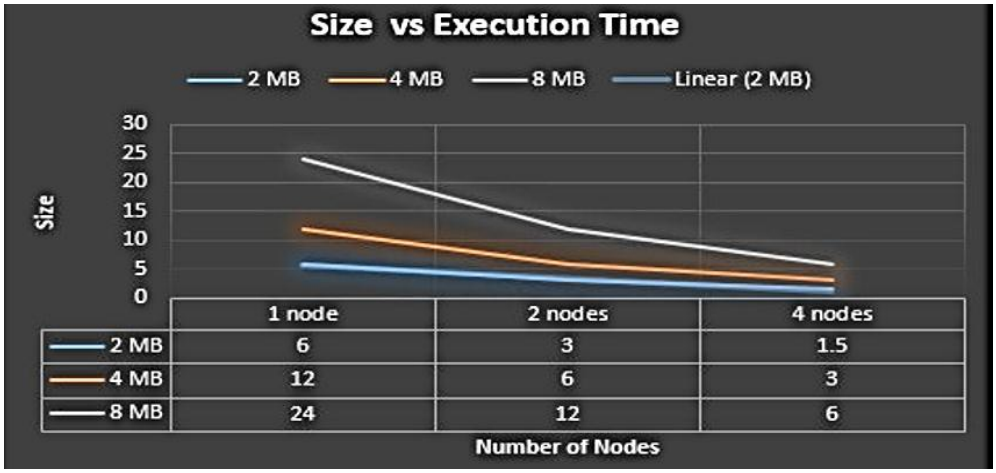

Fig 6: Execution time on Apache Spark based system(s)

If the same analysis needs to be performed without using any technology stack from Big Data ecosystem, the same process will take much more time for similar records.
It shows that calculation on e-governance data using Apache Spark has taken only few seconds to compute and display the name of state that has highest growth in the use of computers among the schools of all the states in India.
Similar big data framework can be implemented easily to analyze e-governance datasets collected from other fields as well.

5. Conclusion

This paper has given the brief introduction of e- governance, its close relationship with big data and how Apache Spark based big data analytic system can be helpful in analyzing government collected datasets accurately at high speed. This research paper gives an insight about Apache Spark framework, its node-based architecture as well as the implementation ways of setting up and analyzing the government collected datasets.
This paper proposes a new system, through an algorithm, to analyze government collected datasets (Elementary Education in India) by setting up a Spark object. This paper further shows the execution time to perform analysis using Spark-shell on single executor; single node Windows based system.
For future references, hypothetical analysis supported by comparison chart has been created to show how execution efficiency of government collected datasets will be increased multiple times using multi-node systems supported by Apache Spark.

6. References

1. Amol Bansod, "Efficient Big Data Analysis with Apache Spark in HDFS, International Journal of Engineering and Advanced Technology", IJEAT. August 2015;4(6).
2. Swapnil Shrivastava, Supriya N Pal, "A Big Data Analytics Framework for Enterprise Service Ecosystems in an e-Governance Scenario", ICEGOV '17, ACM, New Delhi India; c2017.
3. Sruthika S, Tajunisha N. "A study on evolution of data analytics to big data analytics and its research scope", International Conference on Innovations in Information Embedded and Communication Systems, IEEE; c2015.
4. Preet Navdeep, Dr. Manish Arora, Neeraj Sharma, "Role of Big Data Analytics in Analyzing e-Governance Projects", International onference on New Trends in Business and Management: An International Perspective, E-journal ISSN 2250-348X, 2016.
5. Annu kumari, Shailendra Singh, "A review paper on E-governance: transforming government", International Conference on Cloud System and Big Data Engineering, IEEE, 2016, 689-692.7
6. Bhushan Jadhav, Archana B. Patankar, Sonali B. Jadhav, "A Practical approach for integrating Big data Analytics into E-governance using Hadoop", Proceedings of the 2nd International Conference onInventive Communication and Computational Technologies (ICICCT); c2018.
7. http://index-of.co.uk/Big-Data-Technologies/Big%20Data%20Analytics%20with%20R%20and%20 Hadoop.pdf.
8. Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, *et al*., "The rise of "big data" on cloud computing: Review and open research issues", International Journal of Information Sciences, Elsevier; c2014.
9. Hadoop Tutorial, YahooInc., https://developer.yahoo.com/hadoop/tutorial/index.html Big Data Analytics.
10. https://en.wikipedia.org/wiki/Global_Internet_usage.
11. http://worldpopulationreview.com/.
12. http://index-of.co.uk/Big-Data-Technologies/Big%20Data%20Analytics%20with%20R%20and%20 Hadoop.pdf.
13. https://backtobazics.com/big-data/spark/introduction-to-apache- spark/.
14. https://backtobazics.com/big-data/spark/understanding-apache-spark- architecture/.
15. https://www.youtube.com/watch?v=QaoJNXW6SQo.
16. https://www.youtube.com/watch?v=v25NHJJvwVY
17. https://www.youtube.com/watch?v=jffQhcweGwYs
18. https://www.tutorialspoint.com/apache_spark/apache_spark_installati on.htm