ADVANCED ANALYTICAL AUTOMATION MODELING WITH SPARK ON HADOOP FRAMEWORK

Rama Naga Kiran Kumar K [1]*, Prof. I Ramesh Babu [2]*

1. Research Scholar, Dept of CSE, Acharya Nagarjuna University, Nagarjuna Nagar, Guntur, AP. Email: karasala_be@hotmail.com
2. Professor, Dept of CSE, Acharya Nagarjuna University, Nagarjuna Nagar, Guntur, AP.

Abstract: -

Time and Tendency has made the Information Technology to be the market trend, we call as Automation, need in each and everywhere, trending to the Data as the important raw material for the today's world we call it as the Big Data. Hence, in this white paper, the energy and the enthusiastic for the time being given stress on the data used for the energetics decision making, where the entire world moves on. Taking the opportunistic advantage of the Big Data environment, where testing is the biggest challenge for the entire Hadoop or spark or any other framework used to analyze the data to give a realistic picture to the end used, where the decision plays into existence. In this I have given the functional and non-functional deterministic goal driven approach to make the Data scientist and data engineer modelled data. Based on the Modelling, the test condition should be written in the map reduce to know whether the node and function working as expected. Next test-driven approach like to get the optimization and performance like steaming data where spark plays the important role would get the good recommendation. Hence, Big data testing involves the next journey for the optimizations, performance, load balance along with the functional aspect of the data driven by the data scientist needs to be parallel process as the end functional is always a deterministic to the extent of the end user.

1. INTRODUCTION

With respect to software development and verification strategies, testing teams may not yet fully understand the implications of Big Data's impact on the configuration, operation and design of systems and databases. Test Engineers need a clear plan to execute their tests but there are many testers are new unknowns as Big Data systems are layered on top of Enterprise systems struggling with data quality. These challenges are replicating and porting that information into the Big Data analytics and effective software suite.From the business perspective, Big Data is much more important, because the trend model we get from the historical data is quite good in making strategic decision to the decision maker to expand its all the way in the high-end data market, where data is everything. Basically, the importance of software testers understands that Big Data is about far more than simply data volume. Let see an example the oracle database having 2 petabyte data doesn't necessarily constitute a true Big Data situation just a high load. But Big Data management involves fundamentally various methods for storing and processing data, and the required output may also be quite different. With increasing Big Data is imbedded, to the challenges facing the Quality Assurance Testing Departments increases dramatically. This paper presents primer on Big Data testing provides guidelines and methodologies are shows how to approach these data quality problems.

Big Data Characteristics.

Data is the Crucial and the most important part of the Information Industry, leading to the billion of innovation trending the insight view of the data to make strategic decision, where data

scientist, data engineer and more involved. Considering the data , IBM characterize broadly on Four V(s), Volume, Variety, Velocity and Veracity; Volume of data is increasing with @of the million of Terabyte, which earlier time we call as waste data, pointing to the variety of the data, its service of the data collection is different like web logs, data base logs etc, velocity which is major crucial to implement our own processing and give a cute glimpse to decision maker based on the data, hence this the crucial part of the data characterization. Veracity is biases, noise and abnormality in data.

Fig.1. Characteristics of Big Data

With time the data volume is growing exponentially earlier we used to talk about Megabytes or Gigabytes. But time has arrived when we talk about data volume in terms of Terabytes, Petabytes and Zettabytes. Global data volume was around 7.9ZB in 2015 and expected to be 11ZB in 2017. It also known that the global information doubles in every two years.

Again, it comes as the data structure whether it is structured data, unstructured data, or semi-structured data whether we can process the both categories of data will discuss in the functionality part.

The big data testing approach involves both the functional and nonfunctional methodologies. The functional testing performs validating both the quality of the data itself and test environment management ensures that data from a variety of sources is of sufficient quality for accurate analysis and can be processed without deviations. The test scenarios in data quality include completeness, lack of duplication, correctness and more. Generally, the data processing can be done in three ways interactive, batch, and real-time. All these Big Data testing strategies are based on the ETL (Extraction, Transformation and Load) process. Apart from functional testing, non-functional testing plays a major role in ensuring the scalability process.

History of Big Data:

Business intelligence is the concept give rise to Big data. USA has used the data centers to tax data in 1965. After then this started to expand, and the process data visualization started, which become the innovation key. In 2010, Smith from Google announced the getting terabytes of data in every two days, earlier top which given the big data table concept leads to the next level of the concept framework like Hadoop which is the world's first and best reluctant framework to deal with the huge Petabytes of the data, gives rise to other framework like Spark but vase to is the Hadoop.

2. Solution of the Big Data with Hadoop Framework

History of Hadoop and About Core Components Description

Hadoop was created by Doug Cutting Actually Hadoop was the name that came from the imagination of Doug Cutting's son; it was the name that the lies behind his son toy. Brief Chronology of Hadoop Framework.

- 2003 - Google launches project search engine to handle billions of searches and indexing millions of web pages.
- Oct 2003 - Google releases papers with GFS (Google File System)
- Dec 2004 - Google releases papers with Map Reduce
- 2005 - Nutch used GFS and Map Reduce to perform operations
- 2006 - Yahoo! created Hadoop based on GFS and Map Reduce (with Doug Cutting and team)
- 2007 - Yahoo started using Hadoop on a 1000 node cluster
- Jan 2008 - Apache took over Hadoop
- Jul 2008 – Yahoo ran 4000 node Hadoop cluster and Hadoop won terabytes sort benchmark.
- 2009 - Hadoop successfully sorted a petabyte of data in less than 17 hours to handle billions of searches and indexing millions of web pages. And Facebook lunched SQL support for Hadoop.
- Dec 2011 - Hadoop releases version 1.0
- Aug 2013 - Version 2.0.6 is available

3. Hadoop Eco Systems and Functionality

Hadoop ecosystem is the extremely widely adopted and base linear to the every big data ecosystem to preview the as explicit entities are the holistic view of Hadoop architecture gives the glimpse how much robust and makes the other big data framework or tool to do the i/o operation and to visualize the data from decision perspective prominence to Hadoop common, Hadoop 2.x have three major components, these are Hadoop Distributed File Systems (HDFS), Hadoop Map Reduce And YARN (Yet Another Resource Negotiator) also called MapReduce2.0.. Here HDFS is basically used to store large data sets, and MapReduce is used to process such large data sets. YARN is most powerful technology in Hadoop 2.x. unlike 1.x. job Tracker, resource manager and job scheduling/monitoring done in separate daemons. Yarn is a Layer that separate Resource Manager and Node Manager. Hadoop common provides all java libraries, utilities, OS level abstraction, necessary java files and script to run Hadoop. HDFS in Hadoop architecture provides high throughput access to application data and Hadoop Map Reduce provides YARN based Resource Management and parallel processing of large data sets.

The Hadoop Ecosystem comprises of 4 core components –

1) Hadoop Common-

Like Java, where its language we call as technology, similarly to interact with Hadoop ecosystem need to go with the Java archives (JAR files) that are stored in Hadoop Common. Example if we want to use HBase and Hive want to access HDFS they need to make of Java archives (JAR files) that are stored in Hadoop Common.

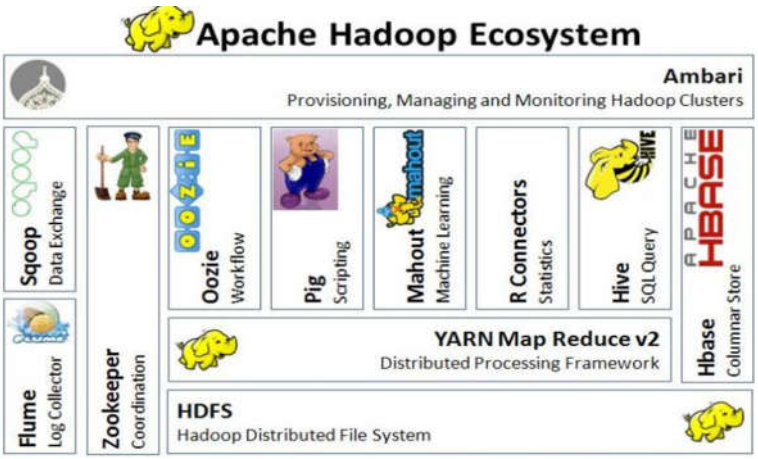


Fig.2. Big Data Eco-System- Apache Hadoop

2) Hadoop Distributed File System (HDFS) -

HDFS is a Java based distributed file system that provides scalable reliable, high-throughput access to the application data stored, across commodity servers. HDFS is highly fault tolerant, where users can dump huge datasets into HDFS and the data will be processed to HDFS component creates several replicas of the data block to be distributed across different servers for reliable and quick data access. HDFS comprises of 4 Daemons -Name Node, Data Node, Job Tracker, and Task Tracker. HDFS operates on a Master-Slave architecture model where the Name Node acts as the master node for keeping a track of the storage cluster and the Data Node acts as a slave node summing up to the various systems within a Hadoop cluster. HDFS also contributes security features to Hadoop. HDFS include file and directory permissions, access control lists, and transparent data encryption.

3) MapReduce-

Distributed Data Processing Framework of Apache Hadoop

In the Distributed data-based framework id the Hadoop Map Reduce is a Java-based system created by Google where the actual data from the HDFS store gets processed efficiently. Map Reduce breaks down a big data processing job into smaller tasks. Map Reduce is responsible for the analyzing large datasets in parallel before reducing it to find the results. In the Hadoop ecosystem, Hadoop Map Reduce is a framework based on YARN architecture. YARN based Hadoop architecture, supports parallel processing of huge data sets and Map Reduce provides the framework for processing the data making distributed across the thousands of nodes, which is highly optimized and have own intelligence to insets the load balance like increasing the node.

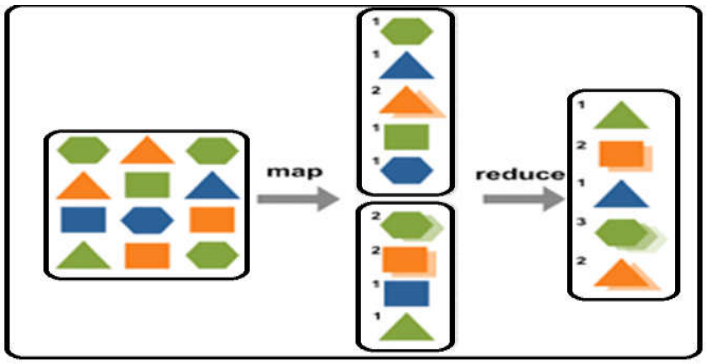


Fig. 3. MapReduce functionality

Archrival Principle in MapReduce is that the Map job sends a query for processing to various nodes in a Hadoop cluster and the Reduce job collects all the results to output into a single value, put data and splits into independent chunks and output of this task will be the input for Reduce Task. Reduce task combines Mapped data tuples into smaller set of tuples. Meanwhile, both input and output of tasks are stored in a file system. MapReduce takes care of scheduling jobs, monitoring jobs and re-executes the failed task. MapReduce forms the compute node while the HDFS file system forms the data node ecosystem architecture both data node and compute node are the same.

4) YARN

Cluster Manager is the integral part of Hadoop 2.x. YARN is great enabler for dynamic resource utilization on Hadoop framework.

Processing tools / framework used for Hadoop

Different vendors came up with their own tool based on input type required for HDFS.

Hadoop Eco System Common Components are HIVE, pig, HBase, SQOOP, flume, Zeppelin, Impala

Hive-

Originally developed by Facebook, Hive is a data warehouse infrastructure built on top of Hadoop. Hive provides a simple, SQL-like language called Hive-QL, whilst maintaining full support for Map Reduce where SQL programmers with little former experience with Hadoop provides indexes, making querying faster. can use the system easier and provides better integration with certain analytics packages like Tableau.

pig -

Analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, enables them to handle very large data sets.

HBase-

Is a NoSQL columnar database which is designed to run on top of HDFS? It is modelled after Google's Big-Table and written in Java capabilities to Hadoop, such as the columnar data storage model and storage for sparse data.

Flume- Flume collects (typically log) data from agent which it then aggregates and moves into Hadoop.

SQOOP- SQOOP is a tool which aids in transitioning data from other database systems (such as relational databases) into Hadoop.

Zeppelin-

It is incubating multi-purposed web-based notebook which brings data ingestion, data exploration, visualization, sharing and collaboration features to Hadoop and Spark.

Impala-

SQL query engine for data stored in a computer cluster running Apache Hadoop provided by Cloudera.

The two key areas for testing problem is

1. Establishing efficient test datasets.
2. Availability of Hadoop centric testing tools such as Hive UDF testing, Pig Unit, Junit for Pig, and Bee Test for Hive.

The major testing should be including all the four phases are show in fig. the data quality issues can be manifest themselves at any of these stages.

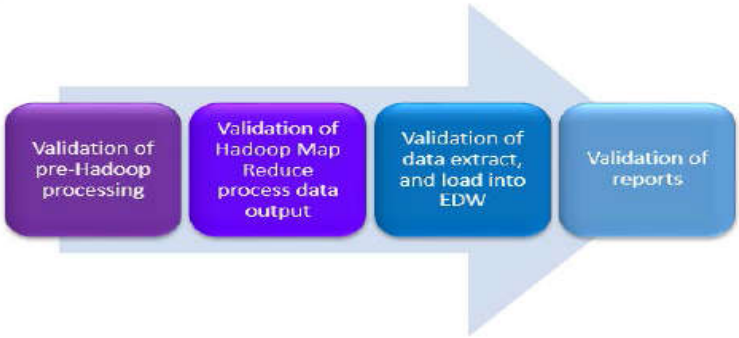


Fig.4. Big Data Testing stages

Big Data testing can be divided into three major steps:

Big Data Staging Validation:

This will first refer as pre-Hadoop stage involves process validation.

Fetching data from various sources like RDBMS, Social media, Weblogs, etc. it should be validating to make sure that accurate data is pulled into system. And comparing the source data with the data pushed into the Hadoop system to make sure the data should match. Then verifying the exact data is extracted and then loaded into the specified HDFS location.

MapReduce Validation:

The second stage is validation of Map Reduce, here the tester verifies the business logic validation on each node in the cluster and then validating them, after running against the multiple nodes, are ensuring the

- ✓ Correctly works the MapReduce process the data
- ✓ On the data aggregation or segregation rules are implemented
- ✓ The processes generated key value pair
- ✓ After completion of MapReduce process, we can validate the data

Output Phase:

The third stage of Big Data testing is validating the output. The generated output data files are ready to move to an EDW (Enterprise Data Warehouse) or any other system based on end user requirements.

In this third stage few activities are includes:

- ✓ Check the applied transformation rules are correctly.
- ✓ Check the data integrity connection and data loaded into the target system successfully.
- ✓ Check that there is no data corruption when we are comparing the target data with the HDFS file system data.

The final stage is reporting on Analytics:

The proposed Big-Data analytics process of organizing, collecting and analyzing the large data sets to patterns and reporting the accurate information.

4. Optimization of Hadoop MR Using Spark with Scala Architecture.

Keeping in mind, methodological, and design challenge in building such a complex and abstract system is to ensure that data is always used for the benefit of all our customers, warful static typing and compile-time checking, and robust frameworks for test-driven development and property-based testing where Scala as Functional programming language gives the best solution.

The Hadoop framework is to processes very large volumes of data for highly resource intensive. In a Application the crucial part is architecture testing is to ensure success of our Big Data project. If the architecture is poorly designed system will impact on the performance, may lead to degrade, and most of the chances are there the system fail to meet the end user requirements. Minimum the failover test and performance test services should be cone in Hadoop Environment.

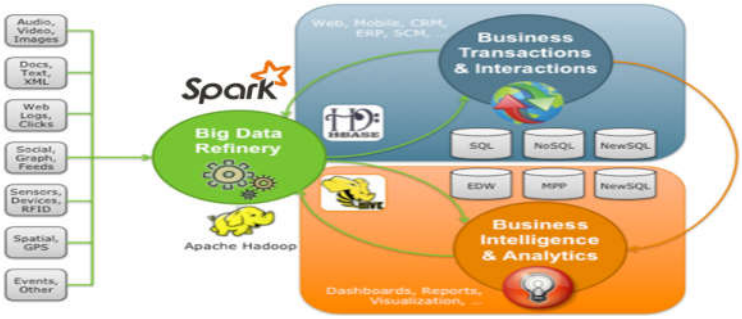


Fig.5. Next generation of Hadoop Data Architecture with Spark

Apache spark is an open source, general purpose processing engine that allows data scientists to build and run fast and sophisticated applications on Hadoop. Spark provides a set of simple and easy-to-understand programming APIs that are used to build applications at a rapid pace in Scala. The Spark Engine supports a set of high-level tools that support SQL like queries, streaming data applications, complex analytics such as machine learning, and graph algorithms.

The state and mutation in Scala makes the evaluation of expressions, especially in parallel systems, much easier than imperative languages like Java or Python leads for harnessing modern hardware that has more cores, more nodes, and more RAM, rather than faster clock speeds. It has the powerful property-based testing DSL to write an exhaustive test suite. In PBT we define properties, not cases. (We just tell the framework that our data packing and unpacking functions need to compose to produce the identity function and Scala Check does the work of finding edge cases, counter examples, off-by-one errors, and so on confidence regarding near-absolute correctness. Spark's rich API facilitates writing MapReduce stages as elegant linear, short code doesn't mean short run times, which is where functional programming, particularly thinking in terms of algebraic types.

```
Val data: RDD[Account] = ...
data. Map (Account => (Account. Deposit, (person. ifsccode, 1)))
. reduce By Key (_ |+| _)
. map Values {case (total, balance) =>
total. to Double / balance
}
. collect ()
```

Large code bases require static typing to eliminate trivial mistakes like Account instead of 12 digits instantly, Complex code requires transparent APIs to communicate design with the Data Frames in a speed of the 100x with compared to Hadoop as two-step MapReduce function in certain applications. Allows data to loaded in-memory and queried repeatedly, making it particularly apt for machine learning algorithms by encapsulating state via OOP and using map Partitions and combine By Key.

Flexibility and Scala features are required to build functionality quickly.

Advantage of Spark

Primary Memory based Processing enables 100x faster
Speed and simplicity
@2X Speedup in the Data node

Uses Functional Programming as the API, reduces the burden of the Function call and the OOPS pros.

Real time Analytics Dashboard Application for Retail:

RADAR is a software solution for retailers utilizing Hadoop technologies including HDFS, YARN, Apache Storm, Apache Solr, Oozie and Zookeeper to help them maximize sales through data based continuous re-pricing.

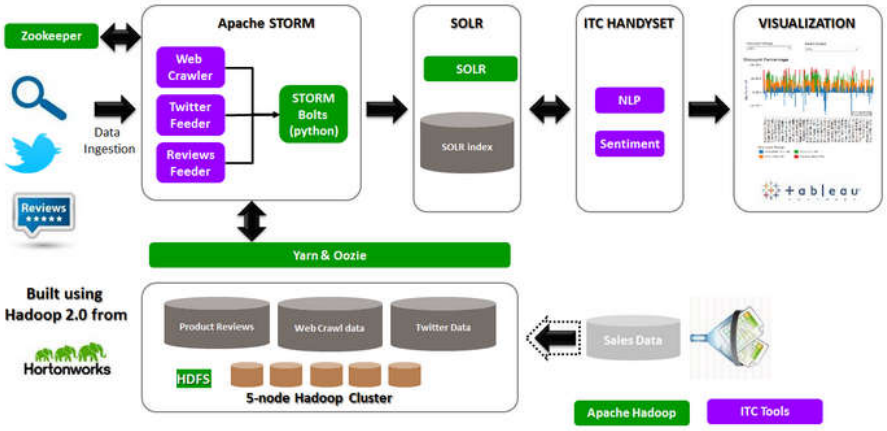


Fig.6. High level Architecture diagram

Specifically using RADAR, a brick and mortar or online retailers can track the following for any number of products in their portfolio:

- Social Sentiment for each product
- Competitive pricing/promotions being offered in social media and on the web.

Using this retailer can crate continuous re-pricing campaigns that can be implemented in real-time in their pricing systems. RADAR can then further track the impact of re-pricing on sales and continuously dashboard it vs social sentiment.

This RADAR tool can also be exposed to retailer's customers are a way to show end customers social sentiment alongside comparative reviews for products being sold by retailer thereby helping to close sales faster as user gets more information to take a decision.

In this architecture we used HDP 2.3.2. For Real-time Analytics Dashboard Application for Retail (RADAR), showcases Apache Storm for real time processing of data and Solr for indexing and data analysis.

For this case a data set of about 1500 TV models was extracted and natural languages text in the data associated with each model analyzed using this engine. Data for each television model was gathered from a, very large number of user's reviews and web pages. Tweets on the other hand are processed real time to extract relevant deals about televisions.

Another show case for Earth Quake Analysis:

Input consists of earthquake data across the world from 1973 to 2010.

Basic attributes like date, magnitude and location are given in the below format.

Year	Month	Day	Time	Latitude	Longitude	Magnitude	Depth
1973	1	1	34609.8	-9.21	150.63	5.3	41
1973	1	1	52229.8	-15.01	-173.96	5	33
1973	1	1	114237.5	-35.51	-16.21	6	33
1973	1	2	5320.3	-9.85	117.43	5.5	66
1973	1	2	22709.2	1.03	126.21	5.4	61
1973	1	2	34752.5	5.4	-82.54	5.2	30
1973	1	2	222557	31.24	88.09	5.2	33
1973	1	3	25816.7	-27.72	-63.26	5.6	563
1973	1	3	143104.5	39.11	71.89	5.5	33
1973	1	3	211416.4	44.34	-129.15	5.3	18
1973	1	4	10750.3	-13.38	167.09	5.1	194
1973	1	4	80350.4	71.11	-7.67	5.1	33
1973	1	4	91032.4	-59.37	-17.89	5.4	33

The input Data file has around 54,000 odd records, and Additionally , the earthquakes are classified on severity of the magnitude as per below format

Severity	Operator	Magnitude
unnoticeable	<	3
Weak	=	3
Light	=	4
Moderate	=	5
Strong	=	6
Severe	=>	7

The input Data file is matched for the severity in this Severity file to produce a composite output

Transformation and Analysis-Results

1)Find the biggest earthquake

Result:(2004,12,26,5853.45,3.3,95.98,9.0,30.0,26-12-2004,)

Note: Severity is not shown as we could not check severity mapping above 7.

2)Find the year with most quakes above 5.0

Result:(2270,2007)

3)Analyse the Earthquakes based on depth-Deepest

Result: (1985,10,22,191401.63, -20.16, -179.16,5.5,700.0,22-10-1985, Moderate)

4)Day of the year when its most likely to have earthquake (>5.0, > 8.0)

Result for > 5: (1656,23)

Repeat same for 8 Magnitude

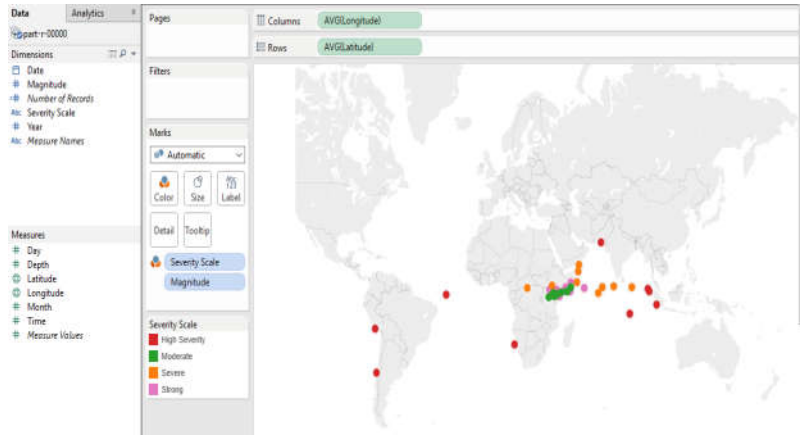
5)Most earthquake prone month (>5.0, > 8.0)

Result:(5391,12)

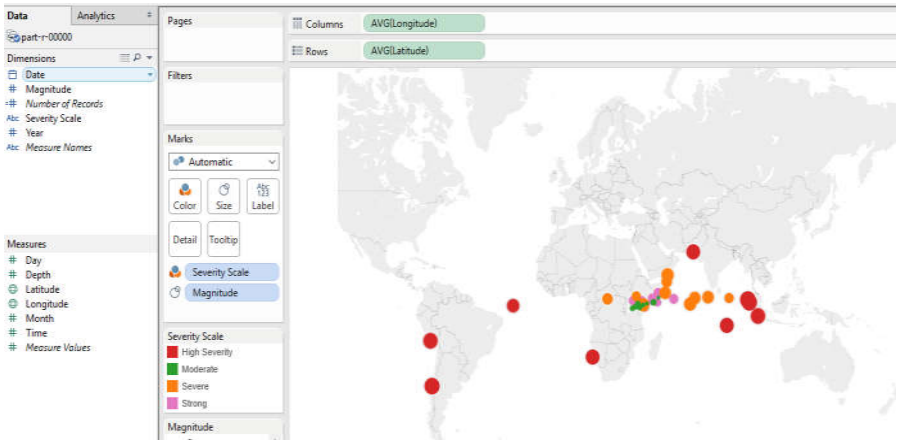
Repeat same for 8 Magnitude

Visualization

Distribution based on Color on Magnitude



Distribution based on size and color on Magnitude



Filters on Year and Severity

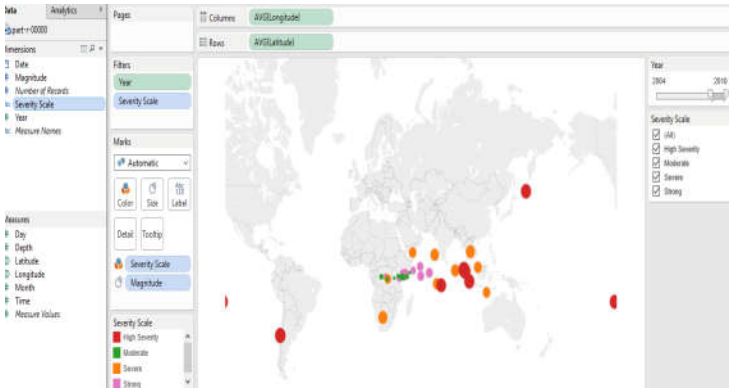
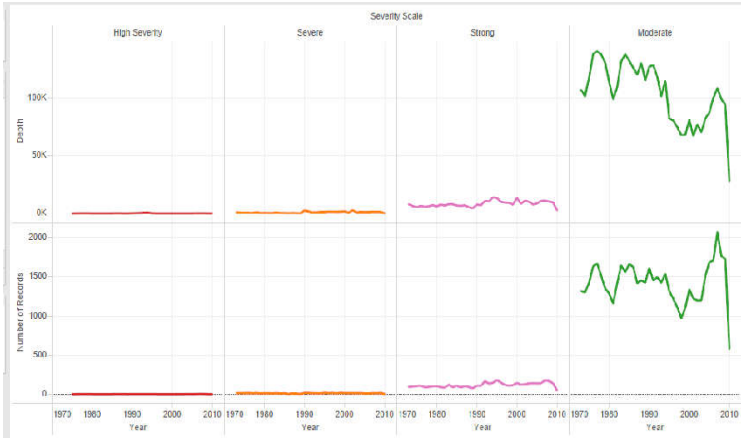


Chart of Severity and Depth per Year for different magnitudes.



Count Analysis on Magnitude

Severity Scale	
High Severity	29
Moderate	54,219
Severe	472
Strong	4,489

CONCLUSION

This paper proposed an Emerging technology of Big Data have paved the way for some exciting new opportunities and also helped people break new ground like never before. This technology being captured data, aggregated and processed very fast rates, with the use of not so expensive hardware. Hadoop is one of the big success story which can be use clusters of commodity hardware to provide scalable, and fast processing of Big Data. Here we propose a parallel programming model for processing and analysis of large scale Remotly sensed data by using Spark on a Hadoop YARN platform. The trend of businesses are take full adavantage of the potential of Big Data, is an effective validation strategy is key. A successes data testing project is for a clear test strategy. This testing strategy can not change the Big Data environment, day to day the complexity has been increased due to the arrival of structured/semi-structured/ unstructured data. An automation testing can give a decisive edge on this complex data. The business success rate is directly proportinal to the effectiveness of an independent testing teams can process and 4Vs to validate all data touch points like structured and unstructured data of large volume, high volume, wide variety and value of the data. With this Hadoop framework the business befits are to avoid the bad business decision, reduce time to market, seamless integration, and reduce total cost of quality. Our approach is not only helps for organizations to build a validation framework over their available data warehouse, but also empowers them and adopt a new Big Data approach for faster data analysis in real time. A successes data testing project is for a clear test strategy.

References:

[1] Chen, M., Mao, S., & Liu, Y, "Big data: A survey", Mobile Networks and Applications Springer, volume 19, issue 2, April2014, pp. 171-209.

[2] Sagiroglu, S., & Sinanc, D, "Big data: A review", IEEE International Conference on Collaboration Technologies and Systems (CTS), 2013, pp 42-47.

[3] Pal, A., & Agrawal, S "An experimental approach towards big data for analyzing memory utilization on a Hadoop cluster using HDFS and MapReduce", IEEE, First International Conference on Networks & Soft Computing (ICNSC), August 2014, pp.442-447.

[4] Zhang, J., & Huang, M. L., "5Ws model for bigdata analysis and visualization," IEEE 16th International Conference on Computational Science and Engineering, 2013, pp.1021-1028.

[5] Qureshi, S. R., & Gupta, A, "Towards efficient Big Data and data analytics: A review", IEEE International Conference on IT in Business, Industry and Government (CSIBIG),March 2014 pp-1-6.

[6] Aravinth, M. S., Shanmugapriyaa, M. S., Sowmya, M. S., & Arun, "An Efficient HADOOP Frameworks SQOOP and Ambari for Big

Data Processing," International Journal for Innovative Research in Science and Technology, 2015, pp. 252-255.

[7] Cloudera- http://www.cloudera.com

[8] http://www.zetta.net/blog/cloud-storage-explained-yahoo

[9] Tang, Z., Jiang, L., Zhou, J., Li, K., & Li, K, "A self-adaptive scheduling algorithm for reduce start time" Future Generation Computer Systems, Elsevier, 2015, pp:51-60.

[10] Zheng, Z., Zhu, J., & Lyu, M. R, "ervice-generated big data and big data-as-a-service: an overview," IEEE International Congress on Big Data (BigData Congress), 2013, pp: 403-410.

[11] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *ACM SIGOPS Operating System Review*. Bolton Landing, New York, USA, 2003, pp. 29–43.

[12] J. Dean and S. Ghemawat, "MapReduce: A flexible data processing tool," *Commun. ACM*, vol. 53, pp. 72–77, 2010.

[13] A. Toshniwal *et al.*, "Storm@twitter," *presented at the Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, Snowbird, UT, USA, 2014.

[14] B. Saha, H. Shah, S. Seth, G. Vijayaraghavan, A. Murthy, and C. Curino, "Apache Tez: A unifying framework for modeling and building data processing applications," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 1357–1369.

[15] K. Wang *et al.*, "Overcoming hadoop scaling limitations through distributed task execution," in *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER'15)*, 2015, pp. 236–245.

[16] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia, *Learning Spark:Lightning-Fast Big Data Analysis*. Sebastopol, CA, USA: O'Reilly Media, Inc, 2015.

[17] E. Vermote, S. Kotchenova, and J. Ray, "MODIS surface reflectance user's guide version 1.3," in *MODIS Land Surface Reflectance Science Computing Facility*, 2011 [Online]. Available: http://www.modissr.ltdri.org/.

[18] Z. Wan, "MODIS land surface temperature products users' guide," Inst. Comput. Earth Syst. Sci., Univ. California, Santa Barbara, CA, USA, 2006 [Online]. Available: http://www.icess. ucsb.edu/modis/LstUsrGuide/usrguide. html.

[19] Y. Qu, Q. Liu, S. Liang, L. Wang, N. Liu, and S. Liu, "Direct-estimation algorithm for mapping daily land-surface broadband albedo from MODIS data," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 2, pp. 907–919, Feb.2014.

[20] Sitthapon Pumpichet, Niki Pissinou, Xinyu Jin and Deng Pan, "Belief-based Cleaning in Trajectory Sensor Streams", IEEE ICCC 2012, Adhoc and Sensor Networking Symposium Pages: 208 - 212, 2012.
[21] [Available online: 14110/2014, 2312] https:llearth.esa.inti
[22] [Available online: 1511012014, 0333] http://www.brockmann-consult.de/cms/web/beam/
[23] Olson, Mike. "Hadoop: Scalable, flexible data storage and analysis." IQT Quarterly 1.3 (2010): 14-18.
[24] Castro P S, Zhang D, Li S. Urban traffic modelling and prediction using large scale taxi GPS traces[M]//Springer, 2012:57-72.
[25] J. D, S. G. MapReduce: simplified data processing on large clusters: Operating Systems Design and Implementation, 2004[C].
[26] Liu L, Andris C, Ratti C. Uncovering cabdrivers' behavior patterns from their digital traces[J]. Computers, Environment and Urban Systems, 2010,34(6):541-548.
[27] Liu Y, Liu X, Gao S et al. Social sensing: a new approach to understanding our socioeconomic environments[J]. Annals of the Association of American Geographers, 2015,105(3):512-530.
[28] Simoes J, Gimènez R, Planagumà M. Big Data y Bases de Datos Espaciales: una análisis comparativo[J]. 2015.
[29] Wang Y, Liu Z, Liao H et al. Improving the performance of GIS polygon overlay computation with MapReduce for spatial big data processing[J]. Cluster Computing, 2015,18(2):507-516.