Time Efficient Dynamic Processing of Big Data for Remote Sensing Application

Mr. Vikas Dudhe
M.Tech Computer Science & Engineering
Nagpur Institute of Technology
Nagpur(Mah),India
dudhevikas7066@gmail.com

Prof. Gunjan Agre
Asst. Professor, M.Tech Computer Science and Engineering
Nagpur Institute of Technology
Nagpur(Mah),India
gunjan.agre@gmail.com

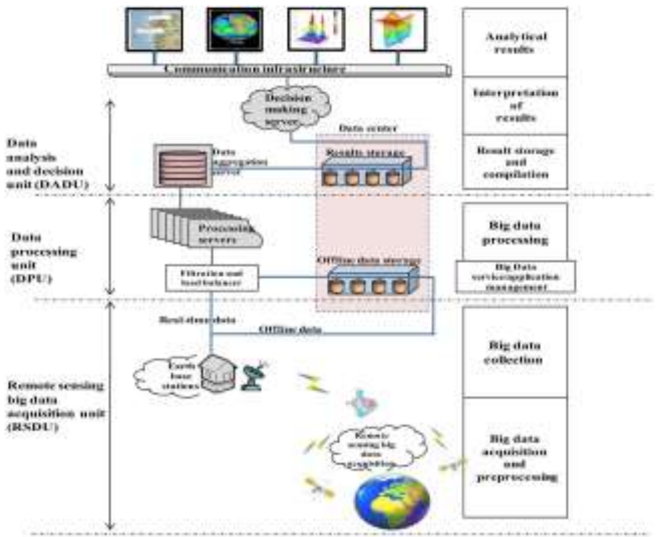
Abstract—Searching info on the web in today's world can be considered as dragging a net across the surface of the earth. While a great deal may be caught in the net, there is still a huge amount of information that is deep, and therefore, missed. The reason is simple: Most of the Web's information is buried down on dynamically produced sites, and standard search engines never find it, where data are hidden behind query interfaces. But a direct query is a "one at a time" laborious way to find info.Several factors contribute to making this problem particularly challenging. The Web is changing at a constant pace – new sources are added, and old sources are removed and modified. The remote wireless senses generate very massive amount real-time data from the Satellite or from the Aircraft with the assistance of the sensors. Technology trends for Big Data accept open source software, commodity servers, and massively parallel-distributed processing platforms. Analytics is at the core of exploiting values from Big Data to produce consumable insights for business and government. This paper presents architecture for Big Data Analytics and explores Big Data technologies offering SQL databases, Hadoop Distributed File System and Map-Reduce. The intended architecture has the aptness of storing incoming unprepared data to dispatch offline analysis on largely stored dumps when required. Concluding, a detailed analysis of remotely sensed earth observatory Big Data for ground or sea level are offered using Hadoop. The proposed architecture possess the ability of dividing, load balancing, and parallel processing of only useful data. Thus, it results in efficient analysis of real-time remote sensing Big Data using earth observatory system.

Keywords*-Big Data, Hadoop, HDFS, Computing operation, RDBMS ,Data Analysis, GFS,Hbase.)*

*****

I. INTRODUCTION

Data is collected from the remote sensors; these remote sensors generates a very large volume raw data this is also called as data acquisition. The collected data has no meaning in it, the sensor simply collects all the information. So the data need to be processed and filtered to extract the useful information from it. [2]The main challenge in this is the data accuracy, the information that are generated by the remote sensors are not in the correct format for analysis. Now the data need to be extracted to pull the useful or meaningful data and converted into to the structured format for best analysis. Sometimes the data might be not clear or it may be erroneous too. To address the above needs, the architecture is introduced, for the remote sensing big data. This architecture has the capacity to analyze both type of data, offline data as well as real time data. First, the data has to be remotely processed in the readable format of the machine then the useful data is sent to the base station of the earth for the further data processing. The earth base station processes 2 types of data one is offline data and the other is real time streaming data. [3]The offline data are sent to the offline data storage device incorporation of the later usage of data. Where in the real time data, the data is directly processed to filtering and the load balancing server. Filtering extracts the meaningful or useful data from the big data and the load balancing will balance processing by distributing the real time data equally to the server. These filtering and the load balancing server will also improve the system efficiency. Understanding the earth atmosphere or environment requires large volume of information or data gathered from different sources, such as air and water quality monitoring sensors, amount of oxygen, co2 and the other gases present in the air, remote access satellite for the observing the characteristics of the earth and so on. In the healthcare scenarios, there is large amount of the data about the agriculture field data contains, CLReportingStatus, Region Territory, Country Crop, Crop detail, SmallholderCluster, SmallholderPercent, AreaSizeMin, AreaSizeAvg, AreaSizeMax, CropSizeMin, CropSizeAvg, CropSizeMaxThe above mentioned data is very complex in nature, there is a chances of missing the importantdata. The challenge is to design a high performance computing systems that can be able integrate resources from different location.[4] Even though the cloud computing systems shown high level performance for RS applications, there are challenges still remaining regarding energy and the time consumption. The big challenge emerges when collecting and the managing Remote Sensing (RS) big data. The RS data are collected from spacecraft, airplanes, satellite and other sensing devices. Remote sensing data growing explosively, we have entered in the period of very high resolution, observation of the earth. Remote sensing data also considered as a "Big Data".


Analytical results
Communication infrastructure
Interpretation of results
Decision making server
Data center
Data analysis and decision unit (DADU)
Data aggregation server
Results storage
Result storage and compilation
Data processing unit (DPU)
Processing servers
Big data processing
Offline data storage
Filtration and load balancer
Big Data service/application management
Real-time data
Offline data
Big data collection
Earth base stations
Remote sensing big data acquisition unit (RSDU)
Remote sensing big data acquisition
Big data acquisition and preprocessing
Fig. 1.Remote sensing Big Data architecture.

II. EXISTING SYSTEM

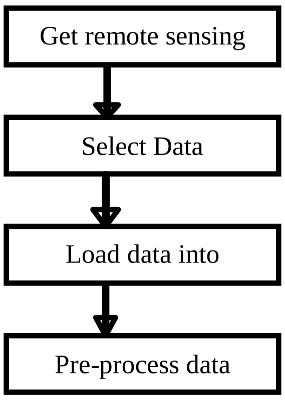
The data stored in the underlying layer of all these technical computing application scenarios have some precise individualities in common, such as 1) large scale data, which refers to the size and the data warehouse; 2) scalability issues, which refer to the application's likely to be running on large scale (e.g., Big Data); 3) sustain extraction transformation loading (ETL) method from low, raw data to well thought-out data up to certain extent; and 4) development of uncomplicated interpretable analytical over Big Data warehouses with a view to deliver an intelligent and momentous knowledge for them [8]. Big Data are usually generated by online transaction, video/audio, email, number of clicks, logs, posts, social network data, scientific data, remote access sensory data, mobile phones, and their applications [6], [7]. These data are accumulated in databases that grow extraordinarily and become complicated to confine, form, store, manage, share, process, analyze, and visualize via typical database software tools. Advancement in Big Data sensing and computer technology revolutionizes the way remote data collected, processed, analyzed, and managed [9]–[12].The incorporation of offline data-storage device helps in later usage of the data, whereas the real-time data is directly transmitted to the filtration and load balance server, where filtration algorithm is designed, which extracts the useful information from the Big Data. On the other hand, the load balancer balances the processing power by equal distribution of the real-time data to the servers. The filtration and load-balancing server not only filters and balances the load, but it is also used to enhance the system efficiency. Furthermore, the filtered data are then processed by the parallel servers and are sent to data aggregation unit (if required, they can store the processed data in the result storage device) for comparison purposes by the decision and analyzing server. The proposed architecture welcomes remote access sensory data as well as direct access network data (e.g., GPRS, 3G, xDSL, or WAN). The proposed architecture and the algorithms are implemented in Hadoop which use MapReduce programming by applying remote sensing earth observatory data.

III. MODULES

- Data Loading and Preprocessing
- Filtration
- Load balancing
- Analysis and Decision

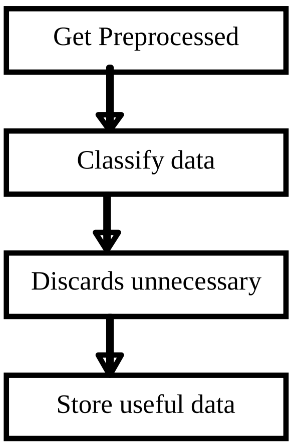
3.1 Data Loading and Preprocessing

• The input of Big Data comes from social networks (Facebook, Twitter, LinkedIn, etc.), Web servers, satellite imagery, sensory data, banking transactions, etc.
• Load remote sensing data into database.
• Pre-process data for remove irrelevant data.



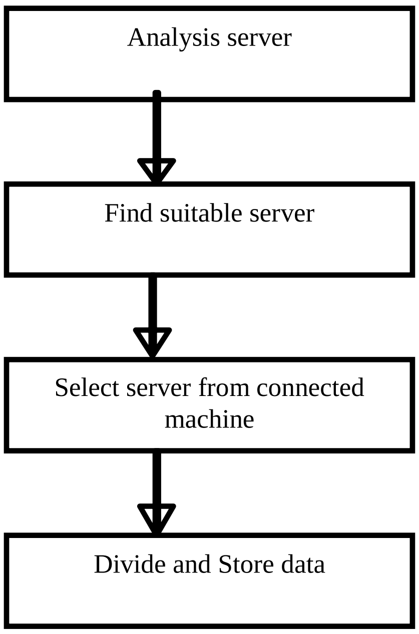
3.2 Filtration

- In data processing unit (DPU), the filtration and load balancer server have two responsibilities, such as filtration of data and load balancing of processing power.
- Filtration identifies the useful data for analysis since it only allows useful information, whereas the rest of the data are blocked and are discarded. Hence, it results in enhancing the performance of the whole proposed system.



3.3 Load balancing

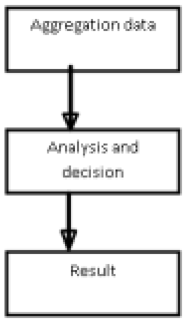
• Apparently, the load-balancing part of the server provides the facility of dividing the whole filtered data into parts and assign them to various processing servers.
• The filtration and load-balancing algorithm varies from analysis to analysis; e.g., if there is only a need for analysis of sea wave and temperature data, the measurement of these described data is filtered out, and is divided into parts.



3.4 Analysis and Decision

• DADU contains three major portions, such as aggregation and compilation server, results storage server(s), and decision making server. [1]When the results are ready for compilation, the processing servers in DPU send the partial results to the aggregation and compilation server, since the aggregated results are not in organized and compiled form.

• Therefore, there is a need to aggregate the results and organized them into a proper form for further processing and to store them. In the proposed architecture, aggregation and compilation server is supported by various algorithms that compile, organize, store, and transmit the results. Again, the algorithm varies from requirement to requirement and depends on the analysis needs.



IV. WORKING

Apache Hadoop is an open-source software framework that supports data-intensive distributed applications, licensed under the Apache v2 license. It supports the running of applications on large clusters of commodity hardware. Hadoop was derived from Google's MapReduce and Google File System (GFS) papers.
The Hadoop framework transparently provides both reliability and data motion to applications. Hadoop implements a computational paradigm named MapReduce, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster. [17]In addition, it provides a distributed file system that stores data on the compute nodes, providing very high aggregate bandwidth across the cluster. Both map/reduce and the distributed file system are designed so that node failures are automatically handled by the framework.
In a larger cluster, the HDFS is managed through a dedicated NameNode server that hosts the file system index, and a secondary NameNode that can generate snapshots of the name node's memory structures, so preventing file system corruption and reducing loss of data. Similarly, job scheduling can be managed by a standalone JobTracker server. In clusters where the HadoopMapReduce engine is deployed against an alternate file system, the NameNode, secondary NameNode and DataNode architecture of HDFS is replaced by the file system-specific equivalent.
In a Hadoop cluster, data is distributed among the nodes of the cluster as it is being loaded in. [15]The Hadoop Distributed File System (HDFS) will split large data files into chunks which are managed by different nodes in the cluster. In addition to this each chunk is replicated across a number of machines, so that a single machine failure does not result in any data being unavailable. An active monitoring system then re-replicates the data in response to system failures which can result in partial storage. Even though the file chunks are copied and distributed across several machines, they form a single namespace, so their contents are universally accessible.
Data is conceptually record-oriented in the Hadoop programming framework. Individual input files are broken into lines or into other formats specific to the application logic. Each process running on a node in the cluster then processes a subset of these records.[8] The Hadoop framework then schedules these processes in proximity to the location of data/records using knowledge from the distributed file system. Since files are spread across the distributed file system as chunks, each compute process running on a node operates on a subset of the data. Which data operated on by a node is chosen based on its locality to the node: most data is read from the local disk straight into the CPU, alleviating strain on network bandwidth and preventing unnecessary network transfers. This strategy of moving computation to the data, instead of moving the data to the computation allows Hadoop to achieve high data locality which in turn results in high performance.

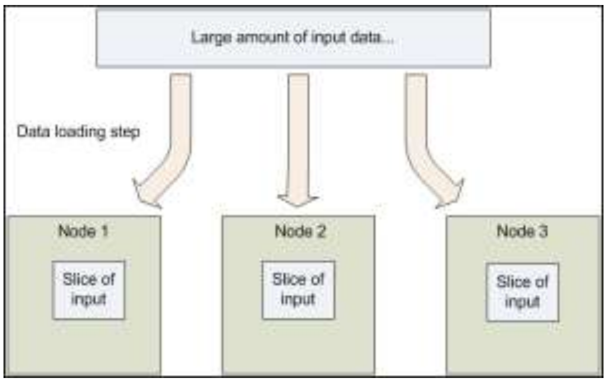


Figure 2: Data distribution in Hadoop

Pseudo code:
K-Means is aeasy learning algorithm for clustering analysis. The goal of K-Means algorithm is to find the best division of n entities in k groups, so that the total distance between the group's members and its corresponding centroid, representative of the group, is minimized.
The k-means algorithm is used for partitioning where each cluster's centre is represented by the mean value of the objects in the cluster.
Pseudo code

```
1. Begin with n clusters, each containing one object and we will number the clusters 1 through n.
2. Compute the between-cluster distance D(p, q) as the between-object distance of the two objects in r and s respectively, p, q =1, 2, …, n. Let the square matrix D = (D(p, q)). If the objects are represented by vectors, we can use the Euclidean distance.
3. Next, find the most similar pair of clusters r and s, such that the distance, D(p, q), is minimum among all the pair wise distances.
4. Merge r and s to a new cluster t and compute the between-cluster distance D(l, m) for any existing cluster m ≠ p, q . Once the distances are obtained, delete the rows and columns corresponding to the existing cluster p and q in the D matrix, since r and s do not exist anymore. Then add a new row and column in D corresponding to cluster l.
5. Repeat Step 3 n − 1 times until there is only one cluster left.
```

k-Means: Step-By-Step Examples easy illustration of a k-means algorithm, consider the following data set consisting of the scores of two variables on each of seven individuals:

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

This data set is to be paired into two clusters. As a first step in finding a sensible initial partition, let the A & B values of the two individuals furthest apart (using the Euclidean distance measure), define the initial cluster means, giving:

	Individual	Mean Vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

The remaining individuals are now examined in sequence and allocated to the cluster to which they are closest, in terms of Euclidean distance to the cluster mean. The mean vector is recalculated each time a new member is added. This leads to the following series of steps:

	Cluster 1		Cluster 2	
Step	Individual	Mean Vector (centroid)	Individual	Mean Vector (centroid)
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

Now the initial partition has changed, and the two clusters at this stage having the following characteristics:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2, 3	(1.8, 2.3)
Cluster 2	4, 5, 6, 7	(4.1, 5.4)

But we cannot yet be sure that each individual has been assigned to the right cluster. So, we compare each individual's distance to its own cluster mean and tothat of the opposite cluster. And we find:

Individual	Distance to mean (centroid) of Cluster 1	Distance to mean (centroid) of Cluster 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

Only individual 3 is nearer to the mean of the opposite cluster (Cluster 2) than its own (Cluster 1). In other words, each individual's distance to its own cluster mean should be smaller that the distance to the other cluster's mean (which is not the case with individual 3). Thus, individual 3 is relocated to Cluster 2 resulting in the new partition:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

The iterative relocation would now continue from new partition until no more relocations occur. However, in this example each individual is now nearer its own cluster mean than that of the other cluster and the iteration stops, choosing the latest partitioning as the final cluster solution.

After years of continuous research and unremitting exploration, Spectral Clustering has been recognized as a clustering algorithm which is more effective than the traditional clustering algorithm, and its mathematical basis is the graph cut and matrix operation. In general, The time complexity of spectral clustering is O (n3), where n is the number of objects to be entered. Because of its high complexity, it greatly limits its application in the actual production and research.

To reduce the time complexity of spectral clustering, this chapter tries to combine spectral clustering algorithm and MapReduce programming ideas of Parallel Spectral Clustering Algorithm Based on HadoopHadoop together. Through the analysis of the traditional spectral clustering algorithm steps, we can achieve steps to separate out and put these steps integration into the MapReduce, combined with Hadoop excellent distributed storage and parallel computing performance, realize the spectral clustering algorithm parallelization, take advantage of the cluster, and reduce the time needed for the clustering ultimately.

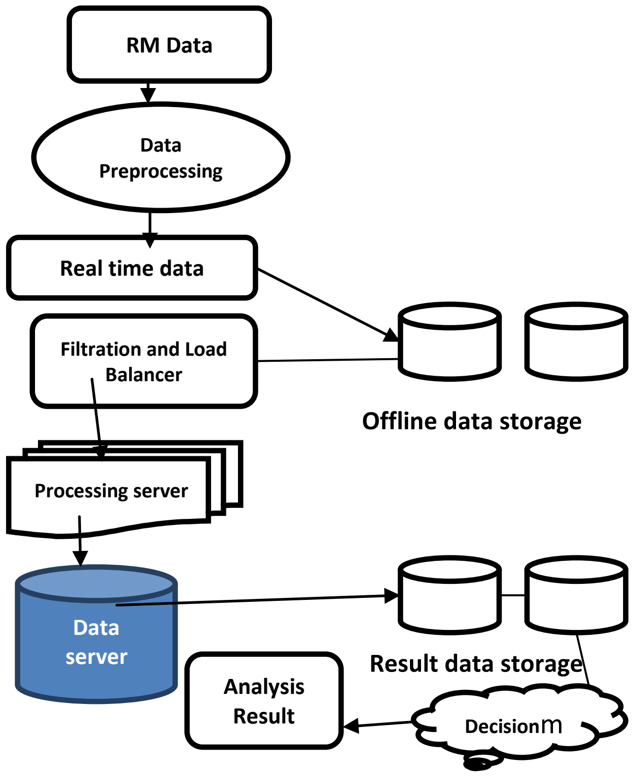


Figure 3: The two-stage architecture of Smart Crawler

V. ALGORITHMS AND NOTATION

5.1 Loading and Data processing

Step1: Calculate the similarity matrix $S\epsilon R^{nxn}$, S(xi,xj) is data points X_i and $X_j$similarity and then sparse it.

Step2 : Constructing diagonal degree matrix D, and diagonal elements are di = $\sum_{j=1}^{n} S(Xi, Xj\ )$

Step3 : Calculate normalized Laplasse matrix L,

$$L = l - D^{\frac{-1}{2}} SD^{\frac{-1}{2}}$$

Step4 : Calculate k the minimum eigenvectors of L, and the composition matrix $Z\epsilon R^{nxk}$ contains them.

Step5 : Standardized Z to $Y\epsilon R^{nxk}$

Step6 : The data points with K-means algorithm $y_i \epsilon R^k$ (i=1,2,…..n) into k clusters C1,…Ck.

Because Hadoop'sMapReduce parallel programming architecture can deliver excellent distributed computing framework, HBase distributed database building on HDFS can be used to initialize and store intermediate results matrix. So, we choose MapReduce, a core component of the Hadoop, to achieve our parallel spectral cluster with the distributed file system HDFS and HBasedistributed database. We first put adjacency matrix which is constitute of the data point 1 2 , , , n x xx into HBase table, the table can be clustered access to aloof the machines, and the key row of each record is set as the index of the datapoints. Then we use a map function to automatically calculate the similarity between the data points.$\forall i, j, 1 \leq i \leq j \leq n,$ we just need to calculate sin(x_i, x_j). Because these objects can constitute undirected graphs sin(x_i, x_j)= sim(x_j, x_i), the calculation of the similarity between each pair of data points needs to be calculated once. And according to the symmetry of undirected graphs, the other half of the similarity values are obtained.

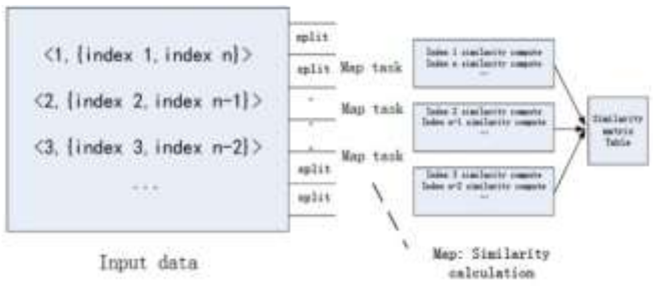


Figure 4: Map Function of parallel computing

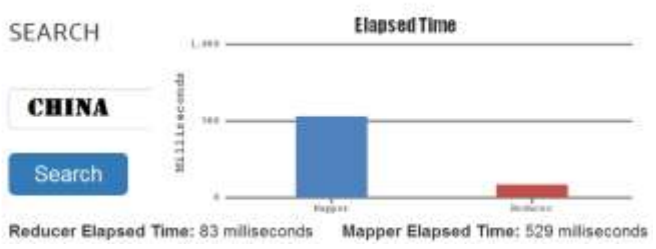
5.2 Filteration and Load Balancing

```
Input: <key, value>, Key as point index, value as null
Step1: index= key, newindex=n-key+1
Step2 : for x in {index,newindex}
x_content =getContentFromHBase(j);
Sim=ComputeSimilarity(x_content,y_content);
storeSimilarity(x,y,sim) into HBase table:
End for
End For
Step3 : Output<key, null>
Step4 : End
```

It should be noted that “similar value of the subscript x” need to calculate the value of n-x+1 pairs of data point {<m_i,m_i>,<m_i,m_{i+1}>…..<m_i,m_n>}. Therefore, in order to load balance, we calculate the similarity of index I and index n-x+1, which performed on the same machine.

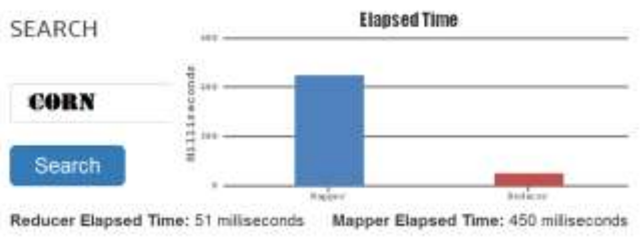
VI. RESULT AND IMPLEMENTATION

We implemented our algorithms in Hadoop environment as it uses java libraries and Hadoop libraries to show the difference between the computational time of searching the contents of earth aggregation information to be processed quickly and shows the comparison on the parameter passed to the system using MapReduce Environment. MapReduce, initially in a single-node environment. In the Hadoop implementation, Map function takes the image block offset as a key and the image block (pixel values) as a value parameter. Since HadoopMapReduce cannot directly process image blocks, the whole product image data are converted into sequence file to be processed using MapReduce. In such a way, one line of the sequence file contains one image block. Map function performs parameters calculations on incoming block values and finally sends the block number as a key and list of parameters results as a value to the Reduce function. Reduce function uses parameter results for performing decision-making on them. We test and evaluate our algorithms with respect to accuracy and processing time using various ESA products [20].



Reducer Elapsed Time: 83 milliseconds Mapper Elapsed Time: 529 milliseconds

Figure 5: Efficiency comparision of HadoopMapReduce implementation and Java Implementation on China key



Reducer Elapsed Time: 51 milliseconds Mapper Elapsed Time: 450 milliseconds

Figure 6: Efficiency comparision of HadoopMapReduce implementation and Java Implementation on corn key

Efficiency measurements are taken by considering the average processing time to process 1-MB data of various products. MapReduce implementation of the analysis algorithm takes less than 1 s the average processing time for various products except Product 3, which takes 1.5 s the average processing time. This processing time among various products varies due to the usage of different bands and image modes, depending on product type. The average processing time for various products is shown in Fig. 8. Finally, a comparison is made between the HadoopMarReduce implementation and the simple Java implementation of the proposed algorithms using average processing time measurements. Hence, for smaller size products, the Hadoop implementation is not efficient because of its lots of input and output operations due to Map and Reduce function. In the case of large-size products, Hadoop divided whole products into blocks and performed parallel tasking on them, which resulted in increased efficiency.

VII. CONCLUSION AND FUTURE DIRECTIONS

In this paper, implemented architecture for real time Big Data analysis for remote sensing application. The proposed architecture efficiently processed offline data initially and further we will extend it to offline data as currently provision is their but it need to sense data dynamically. the capabilities of filtering structured data and processed as per the token passed into the Hadoop System. The algorithm proposed in this paper for each units and subunits are used to analysis remote sensing data sets which helps in better understanding of land agriculture data. In future this system can be used in different datasets with some modifications. In future the system will take the images data and analysis should be done. this algorithms can be used as per the requirements. For Future work we are planning to extend the proposed architecture to make it compatible for Big Data for all applications. Eg sensors and social networking and in data centres. It can further extend to take decision before the natural calamities can be occur and preventive measure can be taken.

VIII. REFERENCES

[1] Real-Time Big Data Analytical Architecture for Remote Sensing Application Muhammad MazharUllahRathore, Anand Paul, Senior Member, IEEE, Awais Ahmad, Student Member, IEEE, Bo-Wei Chen, Member, IEEE, Bormin Huang, and Wen Ji, Member, IEEE.

[2] A. Labrinidis and H. V. Jagadish, “Challenges and opportunities with Big Data,” in Proc. 38th Int. Conf. Very Large Data Bases Endowment, Istanbul, Turkey, Aug. 27–31, 2012, vol. 5, no. 12, pp. 2032–2033.

[3] P. Chandarana and M. Vijayalakshmi, “Big Data analytics frameworks,” in Proc. Int. Conf. Circuits Syst.Commun. Inf. Technol. Appl. (CSCITA), 2014, pp. 430–434.

[4] Wikibon Blog. (Oct. 14, 2014). [2310]. Big Data Statistics [Online].Available: wikibon.org/blog/big-data-statistics/ European Space Agency. (Oct. 14, 2014).

[5] L. Ramaswamy, V. Lawson, and S. V. Gogineni, “Towards a qualitycentric Big Data architecture for federated sensor services,” in Proc. IEEE Int. Congr. Big Data, 2013, pp. 86–93.

[6] X. Li, F. Zhang, and Y. Wang, “Research on Big Data architecture,key technologies, and it’s measures,” in Proc. IEEE 11th Int. Conf. Dependable Auton. Secure Comput., 2013, pp. 1–4.

[7] S. Marchal, X. Jiang, R. State, and T. Engel, “A Big Data architecture for large scale security monitoring,” in Proc. IEEE Int. Congr. Big Data, 2014, pp. 56–63.

[8] X. Yi, F. Liu, J. Liu, and H. Jin, “Building a network highway for Big Data: Architecture and challenges,” IEEE Netw., vol. 28, no. 4, pp. 5–13, Jul./Aug. 2014.

[9] Mr. VikasDudhe, Prof. GunjanAgre “REVIEW PAPER ON REAL TIME BIG DATA FOR MACHINE LEARNING USING HADOOPCLUSTURE”, IJAITE, Vol.2, Issue 3, May-2017 ISSN 2455-6491.

[10] Z. Liu, B. Jiang, and J. Heer, “imMens: Real-time visual querying of Big Data,” Comput. Graph.Forum, vol. 32.no. 3, pp. 421–430, pt. 4, 2013.

[11] European Space Agency. (Oct. 14, 2014). [2312] [Online].Available:https://earth.esa.int/

[12] D. Agrawal, S. Das, and A. E. Abbadi, “Big Data and cloud computing:Current state and future opportunities,” in Proc. Int. Conf. ExtendingDatabase Technol. (EDBT), 2011, pp. 530–533.

[13] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton, “Madskills: New analysis practices for Big Data,” PVLDB, vol. 2, no. 2,pp. 1481–1492, 2009.

[14] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing onlarge clusters,” Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.

[15] H. Herodotou et al., “Starfish: A self-tuning system for Big Data analytics,” in Proc. 5th Int. Conf. Innovative Data Syst. Res. (CIDR), 2011,pp. 261–272.

[16] K. Michael and K. W. Miller, “Big Data: New opportunities and newchallenges [guest editors’ introduction],” IEEE Comput., vol. 46, no. 6,pp. 22–24, Jun. 2013.

[17] Mr. VikasDudhe, Prof. GunjanAgre, “Analytical Architecture of Real Time Big Data for MachineLearning” in Tech-Chronicle International E-journal Tech-ed 2017, ISSN No:2454-1958

[18] C. Eaton, D. Deroos, T. Deutsch, G. Lapis, and P. C. Zikopoulos,Understanding Big Data: Analytics for Enterprise Class Hadoop andStreaming Data. New York, NY, USA: McGraw-Hill, 2012.

[19] R. D. Schneider, Hadoop for Dummies Special Edition. Hoboken, NJ,USA: Wiley, 2012.

[20] M. Mayilvaganan and M. Sabitha, “A cloud-based architecture for Big-Data analytics in smart grid: A proposal,” in Proc. IEEE Int. Conf. Comput. Intell.Comput.Res. (ICCIC), 2013.