International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 4 | ISSN : 2456-3307

Remote Sensing Application with Real-Time Big Data Systematic Construction

N. Sendhil Kumar[1], R. Sravan Kumar Reddy[2], T. Someswara Reddy[2]

[1]HOD, Assoc. Prof, Department of computer applications, Sri Venkateswara College of Engineering and Technology, Chittoor, Andhra Pradesh, India

[2]PG Scholar, Department of computer applications, Sri Venkateswara College of Engineering and Technology, Chittoor, Andhra Pradesh, India

ABSTRACT

In today's era, there is a great deal added to real-time remote sensing Big Data than it seems at first, and extracting the useful information in an efficient manner leads a system toward a major computational challenges, such as to analyze, aggregate, and store, where data are remotely collected. In the existing system we used a new convolution neural network based multimodal disease risk prediction (CNN-MDRP) algorithm using structured and unstructured data. The projected design contains three main units, like 1) remote sensing huge knowledge acquisition unit (RSDU); 2) processing unit (DPU); and 3) knowledge analysis decision unit (DADU). First, RSDU acquires knowledge from the satellite and sends this data to the base Station, wherever initial process takes place. Second, DPU plays an important role in design for efficient processing of period big data by providing filtration, load equalization, and parallel processing. Third, DADU is that the upper layer unit of the projected design, that is accountable for compilation, storage of the results, and generation of call based on the results received from DPU. The projected design has the capability of dividing, load equalization, and parallel processing of only helpful data. Thus, it results in efficiently analyzing real-time remote sensing huge knowledge victimization earth observatory system. Moreover, the projected design has the potential of storing incoming data to perform offline analysis on largely keep dumps, once needed. Finally, detailed analyses of remotely detected earth observatory huge data for land and ocean space are provided using Hadoop. in addition, various algorithms are proposed for every level of RSDU, DPU, and DADU to find land as well as sea area to elaborate the operating of an architecture.

Keywords : Big Data, data analysis decision unit (DADU), data processing unit (DPU), land and sea area, offline, real-time, remote senses, remote sensing Big Data acquisition unit (RSDU).

I. INTRODUCTION

As of late, a lot of enthusiasm for Big Data has risen, for the most part determined from across the board number of research issues emphatically identified with genuine applications and systems. Day by day the data is increasing very large volume from social media, videos, emails, online transitions, logs, scientific data, mobile phones, Remote sensors and other applications. These data store in the database and grow rapidly with a massive amount becomes complicated to store, process, manage and analyze. The advanced technology in the big data gives a way to the remote data, which can be collection, managing, analyzing and processing. Recently designed remote sensors that are used for the earth observatory streams the data continuously and generates large amount of data.

Many of the work have been done in the different fields of remote sensing data from the satellite, such as gradient based edge detection change detection and etc. This paper is concentrated on the high speed continuous real time streaming data or large amount of offline data i.e. big data, this leads to a new challenge. Such consequences for scientific understanding of transformation of the remote sensed data is critical task. Data is collected from the remote sensors; these remote sensors generate 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. 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. 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. Next, the data is directly sent to the data aggregation unit for comparison by analyzing and the decision server. The proposed method is implemented by the Hadoop framework using the map reduce programming by the data of remote sensing.

II. PROPOSED SYSTEM

Algorithm Design and Testing:-

On the basis of the analysis made in the previous section, a set of algorithms is proposed to process high-speed, large amount of real-time remote sensory image data using our proposed architecture. It works on both DPU and DADU by taking data from satellite as input to identify land and sea area from the data set. The set of algorithms contains four simple algorithms, i.e., algorithm I, algorithm II, algorithm III, and algorithm IV that work on filtrations and load balancer, processing servers, aggregation server, and on decision-making server, respectively. Algorithm I, i.e., filtration and load balancer algorithm (FLBA) works on filtration and load balancer to filter only the require data by discarding all other information. It also provides load balancing by dividing the data into fixed size blocks and sending them to the processing server, i.e., one or more distinct blocks to each server. This filtration, dividing, and load-balancing task speeds up our performance by neglecting unnecessary data and by providing parallel processing. Algorithm II, i.e., processing and calculation algorithm (PCA) processes filtered data and is implemented on each processing server. It provides various parameter calculations that are used in the decision-making process. The parameters calculations results are then sent to aggregation server for further processing. Algorithm III, i.e., aggregation and compilations algorithm (ACA) stores, compiles, and organizes the results, which can be used by decision-making server for land and sea area detection. Algorithm IV, i.e., decision-making algorithm (DMA) identifies land area and sea area by comparing the parameters results, i.e., from aggregation servers, with threshold values.

Algorithms parameters and variables:

Following are the parameters and variables used in the proposed algorithms. B1, B2, B3, B4, B5 ... BN is image fixed size blocks. NR: Number of records in MDS (number of lines in the image). NSR: Number of samples in each record (number of pixels in each line). BS: Image block size (i.e., block size of B1, B2,..., BN). N: Total number of blocks in the image.

$$\text{i.e., } N = \frac{NR \times NSR}{BS}.$$

PSB: Processed sample block (PSB = {B1, B3, B5, B7 ... BN–1}). UPSB: Unprocessed sample block (UPSB = {B2, B4, B6, B8 ... BN}).

XBi: Mean of sample values of block Bi, where i = {1, 2, 3, 4 ... N}. XBi= Sum of all values of the block Bi/size of block

$$= \sum_{j=1}^{BS} \frac{Vj}{BS}$$

Vj: jth value in block Bi. 1 ≤ j ≤ BS. SDBi: Standard deviation of sample values of block Bi.

$$\text{SD}_{\text{Bi}} = \sqrt{\frac{\sum_{j=1}^{BS} (V_J - \overline{X}_{Bi})}{BS}}.$$

Abs_Diff: Absolute difference between XBi and SDBi. AbsDiff = XBi – SDBi .

Maxval: Threshold value is set and is greater than normal range to check how many values of the block are deviated from the normal range.

NGmaxval: Number of values in the block is greater than Maxval. Below are the threshold variables, which are set on the basis of analysis, i.e.,

∂X: Mean threshold is set on the basis of analysis, which is used to compare the mean value of each block with threshold for detecting land, sea, or any other area.

∂SD: SD threshold is set on the basis of analysis, which is used to compare the SD value of each block with threshold for detecting land, sea, or any other area.

∂Abs_diff: Absolute difference threshold is set on the basis of analysis, which is used to compare absolute value of each block with threshold for detecting land, sea, or any other area.

∂NGmaxval: Threshold for number of values that are greater than Maxval is set on the basis of analysis, which is used to compare NGmaxval value of each block with threshold for detecting land, sea area, or any other area.

Algorithm I. Filtration and Load Balancing Algorithm (FLBA)

Input: Satellite process data set/product
Output: filtered Image data in fixed size block and send each block to processing server
Steps:
1. Filter Image related data i.e. Processed data in MDS. All other unnecessary data will be discarded.
2. Divide the image into fixed size block i.e. $BS = 100 \times 100$ MDS process_data values, row by row fashion or column by column. Each block will be denoted by B_i where $1 \leq i \leq BS$
3. Make two samples of blocks so that only half of the part is processed.
i.e., $PSB = \{B_1, B_3, B_5, \ldots, B_{N-1}\}$ and $UPSB = \{B_2, B_4, B_6, B_8, \ldots, B_N\}$
4. Transmit UPSB directly to aggregation server without processing.
5. Assign and transmit each distinct block(s) Bi of PSB to various processing servers in DPU.

Description: This algorithm takes satellite data or product and then filters and divides them into segments and performs load-balancing algorithm. In step 1, the image-related data resides in the MDS part of the product, is filtered out. In step 2, filtered data are the association of different numbers of record and each record is different numbers of sample, which results in forming a matrix. The matrix is then divided into fixed sized blocks. A set of building blocks is {B1, B2, B3, B4,..., BN} that contains 100 × 100 values. Block size can be more or less depending on the number of servers and computation power. In step 3, these blocks are divided into two parts, i.e., PSB and UPSB. UPSB is processed by PDU, whereas the decision of UPSB is based on PSB processing results.

Algorithm II. Processing and Calculation Algorithm (PCA)

Input: Block B_i
Output: statistical parameters results and transmit them to aggregation server.
Steps:
1. For each Block B_i, Calculate
a. $\overline{X}$Bi
b. $S.D_{Bi}$
c. Abs_Diff
d. NG_{maxval}
2. Transmit the results against block id and product id to the aggregation server in DADU

Description: The processing algorithm calculates results for different parameters against each incoming block and sends them to the next level. In step 1, the calculation of mean, SD, absolute difference, and the number of values, which are greater than the maximum threshold, are performed. Furthermore, in the next step, the results are transmitted to the aggregation server.

Algorithm III. Aggregation and Compilation Algorithm (ACA)

Input: Block B_i results
Output: compiling, storing and sending PSB results and UPSB blocks information to decision-making server.
Steps:
1. Collect Every B_i's result of PSB
2. Compile them and transmit them to Decision-making server.
3. Store PSB blocks with results and UPSB blocks without result into RBMS in result storage.

Description: ACA collects the results from each processing servers against each Bi and then combines, organizes, and stores these results in RDBMS database. It also stores UPBS information into the database. ACA transmits a copy of PSB results and UPSB information to decision-making server for real-time decision-making.

Algorithm IV. Decision-making algorithm (DMA)

Input: PSB results and UPSB information
Output: each block Bi with decision, land block or sea. Finally, the whole image is divided into sea and land area
Rules:
Following rules are made on the basis of land area analysis discussed in Section III for detecting land block
1. $\overline{X}\ \mathrm{Bi} \leq \partial_{\overline{X}}$
2. $\mathrm{S.D_{Bi}} \geq \partial_{S.D}$
3. $\mathrm{Abs\_Diff} \geq \partial_{Abs\_diff}$
4. $\mathrm{NG_{maxval}} \leq \partial_{NGmaxval}$

Steps:
1. For Each (B_i of PBS)
{
  If ($Rule1 == true\ and\ Rule\ 2 == true$)
  $\mathrm{Status\_B_i} = \mathrm{Land}$
  Else if ($Rule1 == false\ and\ Rule\ 2 == false$)
    $\mathrm{Status\_B_i} = \mathrm{Sea}$
  Else
  {
    If ($Rule3 == false\ and\ Rule\ 4 == false$))
      $\mathrm{Status\_B_i} = \mathrm{Sea}$
    Else
      $\mathrm{Status\_B_i} = \mathrm{Land}$
  }
}
2. For Each (Bi of UPBS)
{
  If ($\mathrm{Status\_B_{i-1}} == \mathrm{Land and\ status\_B_{i+1}} = \mathrm{Land}$)
$\mathrm{Status\_B_i} = \mathrm{Land}$
  Else If ($\mathrm{Status\_B_{i-1}} == \mathrm{Sea and\ status\_B_{i+1}} = \mathrm{Sea}$)
$\mathrm{Status\_B_i} = \mathrm{Sea}$
Else
$\mathrm{Status\_B_i} =! (\mathrm{Status\_B_{i-1}} \oplus \mathrm{status\_B_{i+1}} \oplus \mathrm{status\_B_{i+3}})$
}

Description: DMA takes results of each block of PSB and UPSB information as well. It then analyzes results of each block of PSB and determines whether the block belongs to land or sea. For each block Bi of PSB if XBi ≤ ∂X and SDBi ≥ ∂SD, then the Bi is detected as land block and if XBi ≤ ∂X and SDBi ≥ ∂SD both are false then the block Bi is detected as sea. If XBi ≤ ∂X is false and SDBi ≥ ∂SD is true or vice versa, then the block is tested for rule 3 and 4, if both rules are false, then it is detected as sea and otherwise, it is detected as land.

For UPSB, the block is identified on the basis of neighbor's block results. The neighbor block of each UPSB is in PSB. For every UPSB block Bi, if their neighbor's blocks, i.e., Bi+1, Bi–1 that are PSB blocks, are land, and then Bi is detected as land. If its neighbor blocks are sea, then Bi is detected as sea. For its neighboring blocks, if one is considered as land and other is considered as sea, then we need to find the decision of next neighbor, i.e., Bi+3, Bi is detected as land, if two of these neighbor block are land; otherwise, it will be sea. Finally, when each block of PSB and UPSB is detected as sea or land, then the decision-making process is completed, which uses these decisions in any application or just for announcement or display.

III. CONCLUSION

In this paper, we have a tendency to planned design for period massive data analysis for remote sensing application. The planned architecture efficiently processed and analyzed period and offline remote sensing massive knowledge for decision-making. The planned architecture consists of 3 major units, like 1) RSDU; 2) DPU; and 3) DADU. These units implement algorithms for each level of the design looking on the specified analysis. The design of real-time big is generic (application independent) that's used for any kind of remote sensing massive Data analysis. Furthermore, the capabilities of filtering, dividing, and multiprocessing of solely helpful info

square measure performed by discarding all alternative further knowledge. These processes make a much better alternative for period remote sensing big data analysis. The algorithms planned during this paper for every unit and subunits are used to analyze remote sensing knowledge sets, which helps in higher understanding of land and ocean space. The planned architecture welcomes researchers and organizations for any type of remote sensory big data analysis by developing algorithms for every level of the architecture depending on their analysis requirement.

IV. REFERENCES

[1]. A. Abouzeid, K. B. Pawlikowski, D. J. Abadi, A. Rasin, and A. Silberschatz. HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB, 2(1):922–933, 2009.

[2]. D. Agrawal, S. Das, and A. E. Abbadi. Big data and cloud computing: New wine or just new bottles? PVLDB, 3(2):1647–1648, 2010.

[3]. D. Agrawal, A. El Abbadi, S. Antony, and S. Das. Data Management Challenges in Cloud Computing Infrastructures. In DNIS, pages 1–10, 2010.

[4]. P. Agrawal, A. Silberstein, B. F. Cooper, U. Srivastava, and R. Ramakrishnan. Asynchronous view maintenance for vlsd databases. In SIGMOD Conference, pages 179–192, 2009.

[5]. S. Aulbach, D. Jacobs, A. Kemper, and M. Seibold. A comparison of flexible schemas for software as a service. In SIGMOD, pages 881–888, 2009.

[6]. P. Bernstein, C. Rein, and S. Das. Hyder – A Transactional Record Manager for Shared Flash. In CIDR, 2011.

[7]. M. Brantner, D. Florescu, D. Graf, D. Kossmann, and T. Kraska. Building a database on S3. In SIGMOD, pages 251–264, 2008.

[8]. F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In OSDI, pages 205–218, 2006.

[9]. J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton. Mad skills: New analysis practices for big data. PVLDB, 2(2):1481–1492, 2009.

[10]. B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. PNUTS: Yahoo!'s hosted data serving platform. Proc. VLDB Endow., 1(2):1277–1288, 2008.

[11]. C. Curino, E. Jones, Y. Zhang, E. Wu, and S. Madden. Relational Cloud: The Case for a Database Service. Technical Report 2010-14, CSAIL, MIT, 2010. http://hdl.handle.net/1721.1/52606.

[12]. S. Das, S. Agarwal, D. Agrawal, and A. El Abbadi. ElasTraS: An Elastic, Scalable, and Self Managing Transactional Database for the Cloud. Technical Report 2010-04, CS, UCSB, 2010.

[13]. S. Das, D. Agrawal, and A. El Abbadi. ElasTraS: An Elastic Transactional Data Store in the Cloud. In USENIX Hot Cloud, 2009.

[14]. S. Das, D. Agrawal, and A. El Abbadi. G-Store: A Scalable Data Store for Transactional Multi key Access in the Cloud. In ACM SOCC, 2010.

[15]. S. Das, S. Nishimura, D. Agrawal, and A. El Abbadi. Live Database Migration for Elasticity in a Multitenant Database for Cloud Platforms. Technical Report 2010-09, CS, UCSB, 2010.

[16]. S. Das, Y. Sismanis, K. Beyer, R. Gemulla, P. Haas, and J. McPherson. Ricardo: Integrating R and Hadoop. In SIGMOD, 2010.

[17]. J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. In OSDI, pages 137–150, 2004.

[18]. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In SOSP, pages 205–220, 2007.

[19]. D. J. Dewitt, S. Ghandeharizadeh, D. A. Schneider, A. Bricker, H. I. Hsiao, and R. Rasmussen. The Gamma Database Machine Project. IEEE Trans. on Knowl. and Data Eng., 2(1):44–62, 1990.

[20]. The Apache Hadoop Project. http://hadoop.apache.org/core/, 2009