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

Data Encryption Strategy with Privacy-Preserving for Big Data in Mobile Cloud using H2Hadoop

T. Ramathulasi[1], C. Samba Shiva Reddy[2], K. Ashok Kumar[2]
[1]Asst.Professor, Department of Computer Applications SVCET, Chittoor, Andhra Pradesh, India
[2]PG Scholar, Department of Computer Applications SVCET Chittoor, Andhra Pradesh, India

ABSTRACT

Cloud Computing leverages Hadoop framework for process Big Data in parallel. Hadoop has bound limitations that could be exploited to execute the duty efficiently. These limitations square measure principally thanks to data section within the cluster jobs and tasks scheduling, and resource allocations in Hadoop. Economical resource allocation remains a challenge in Cloud Computing MapReduce platforms. We propose H2Hadoop that is an enhanced Hadoop design that reduces the computation value related to Big Data analysis. The projected design also addresses the difficulty of resource allocation in native Hadoop. H2Hadoop provides a better resolution for "text data", like finding DNA sequence and the motif of a dna sequence. Also, H2Hadoop provides an efficient Data Mining approach for Cloud Computing environments. H2Hadoop architecture leverages on Name Node's ability to assign jobs to the TaskTrakers (Data Nodes) inside the cluster. By adding control options to the Name Node, H2Hadoop will intelligently direct and assign tasks to the Data Nodes that contain the desired knowledge while not causing the duty to the full cluster. Comparing with native Hadoop, H2Hadoop reduces cpu time, range of read operations, and another Hadoop factors.

Keywords: Big Data, Cloud Computing, Hadoop, H2Hadoop, Hadoop Performance, MapReduce, Text Data.

I. INTRODUCTION

Parallel process in Cloud Computing has emerged as an knowledge base analysis space because of the heterogeneous nature and enormous size of information. Translating sequential knowledge to important info needs substantial procedure power and economical algorithms to identify the degree of similarities among multiple sequences. serial pattern mining or knowledge analysis applications like, DNA sequence orientating and motif finding typically need giant and sophisticated amounts of information processing and procedure capabilities . With efficiency targeting and planning of procedure resources is required to unravel such advanced issues. Although, a number of the information sets ar decipherable by humans, it is terribly advanced to be understood and processed using ancient process technique. Handiness of open supply and business Cloud Computing parallel processing platforms have opened new avenues to explore structured, semi-structured or unstructured knowledge. Before we go any longer, it's necessary to outline sure definitions that are associated with Big Data and Hadoop.

Proposed algorithm:-

H2HADOOP:-

In existing Hadoop architecture, Name Node knows the location of the data blocks in HDFS. Name Node is responsible for assigning the jobs to a client and dividing that job into tasks. Name Node further assigns the tasks to the TasTrackers (Data Nodes). Knowing which Data Node holds the blocks containing the required data, Name Node should be able to direct the jobs to the specific Data Nodes without going through the whole cluster. In

H2Hadoop, before assigning tasks to the Data Nodes, we implemented a pre-processing phase in the Name Node. Our focus is on identifying and extracting features to build a metadata table that carries information related to the location of the data blocks with these features. Any job with the same features should only read the data from these specific blocks of the cluster without going through the whole data again. Explanation of the proposed solution is as follows:

Common Job Blocks Table (CJBT):-

Proposed Hadoop MapReduce workflow (H2Hadoop) is the same as the original Hadoop in terms of hardware, network, and nodes. However, the software level has been enhanced. We added features in Name Node that allow it to save specific data in a look up table which named Common Job Blocks Table CJBT. The proposed solution can only be used for text data. Big Data, such as Genomic data and books can be processed efficiently using the proposed framework. CJBT stores information about the jobs and the blocks associated with specific data and features. This enables the related jobs to get the results from specific blocks without checking the entire cluster. Each CJBT is related to only one HDFS data file, which means that there is only one table for each data source file(s) in HDFS. In our research, we took an example of genome Big Data to show the functionality of enhanced Hadoop architecture. In order to understand the framework of Mapping and Reducing in the proposed platform, we searched for a DNA sequence using H2Hadoop in HDFS. Sequence aligning is an essential step for many molecular biology and bioinformatics applications, such as phylogenetic tree construction, gene finding, gene function, and protein structure prediction. Computationally intensive algorithms are used for sequence alignment. Scalable parallel processing Hadoop framework has been proposed and implemented for the sequence alignment of genomic data. Proposed Hadoop architecture relies on CJBT for efficient data analysis. Each time a sequence is aligned using dynamic programming and conventional alignment algorithms, a common feature that is a sequence or subsequence is identified and updated in CJBT. Common features in CJBT can be compared and updated each time clients submit a new job to Hadoop. Consequently, the size of this table should be controlled and limited to a specific size to keep the architecture reliable and efficient. A typical CJBT consists of three main components or columns (Table I), which are explained below:

TABLE I

COMMON JOB BLOCKS TABLE COMPONENTS

Common Job Name	Common Feature	Block Name		
Sequence_Alignment	GGGATTTA	B1	B2	B3
	TTTAGA	B1	B4	
Fining_Sequence	TTTAGCC	B3	B6	
	GCCATTAA	B1	B3	B4
	AATCCAGG	B3	B5	

Common Job Name CJN:-

Common Job Name represents a shared name of a job that each MapReduce client must use when submitting a new job in order to get the benefit of the proposed architecture. We define a library, which contains a list of pre-coded jobs that is made available to the user by an Application Program Interface (API). The Jobs APIs provide a brief job description and access to job data. The users select a job name (or shared database name) from the list of jobs already identified for a shared MapReduce job (or data). This feature helps Name Node to identify and match a job to a Data Node(s) containing block(s) in the CJBT.

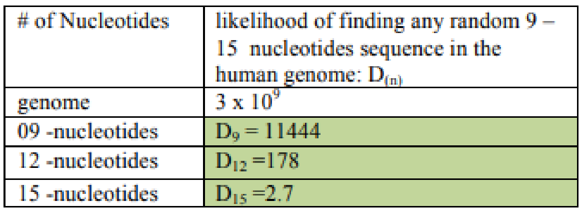
Common Feature CF:-

Common Features are defined as the shared data between jobs. H2Hadoop supports caching, enables output (or part of output) to be written in the CJBT during the reduce step. We use Common Features to identify the Data Nodes or the blocks with shared data entries. JobTracker directs any new jobs with the shared common features to block names in CJBT. Suppose J1 and J2 are sequence search jobs, J1 uses MapReduce to find the sequence in a DataNode or a block. If J2 contains common feature of J1, it is logical to map the task and allocate the same data resources of J1. When a sub-sequence arrives to the Name Node as the result of a new job, the old common feature will be replaced with the old one. However, feature selection should be done carefully as the response time for the jobs can increase if common features exist in every Data Node. For example, in genomic data, regulatory sequences and protein binding sites are highly recurring sequences. Using such sequences as common features can degrade the performance of the proposed solution. The length of common feature also plays on important role in the proposed solution. If the sequence is too short it will be present many times in all chromosomes and all datasets. For a random sequence Dn is the likelihood of how many times a DNA sequence occurs in the whole human genome. The likelihood of the binding sites for 9, 12 and 15 fingers, ZNF is presented in (TABLE II). For a random sequence of length Dn, where n is the length of nucleotide sequence, the likelihood of how many times a sequence occurs in the whole human genome is given by:

$$D_n = 3 \times 10^9 / (4)^n$$

TABLE II

LIKELIHOOD OF RANDOM NUCLEOTIDES



# of Nucleotides	likelihood of finding any random 9 – 15 nucleotides sequence in the human genome: $D_{(n)}$
genome	3×10^9
09 -nucleotides	$D_9 = 11444$
12 -nucleotides	$D_{12} = 178$
15 -nucleotides	$D_{15} = 2.7$

As shown in (TABLE II), the likelihood of any random 9 base pair (bp) of a long nucleotides sequence in a whole genome is quite large comparing with 12 base pair (bp), and using a 9 bp long sequence as a common feature will result in the performance degradation of the proposed architecture. The probability of any random 12 bp long sequence in a human genome is 5.96 x 10-8 equaling 178 times.

Block Name BN:-

BlockName or BlockID is the location of the common features. It identifies the block(s) in a cluster where certain information is stored. BlockName helps the NameNode direct jobs to specific DataNodes that store these blocks in HDFS. CJBT has the list of all blocks that are related to the results of the common feature. For example, if a sequence “TTTAGATCTAAAT” is only stored in B1 and B4, the NameNode will direct any job that has a particular sequence to B1 and B4. This CJBT is a dynamically configurable table and the BlockName entries are changing as the common feature changes. CJBT should not become too large because larger lookup table tends to decrease the system performance. The size of CJBT can be limited by employing the 'leaky bucket' algorithm. The 'leaky bucket' parameters can be adjusted to keep the size of CJBT constant. This can be discussed more in future work.

End-User Interface:-

A user interface gives the user a list of Common Job Names (CJN) to choose from. As the tasks are completed, CJBT is dynamically updated and more relationships are defined. If the CJBT is empty, the user will execute the MapReduce job in a traditional way without getting the benefits of the proposed solution. The predefined CJN and CF are defined either by the user or by the user interface manager, which might become a central source for updating the lists for all clients.

H2Hadoop MapReduce Workflow:-

Enhanced Hadoop architecture doesn’t differ from the native Hadoop architecture so it will be enhancing only the software level through build CJBT. Following chart (Figure 1) shows the proposed changes in Name Node, which works as a lookup table that contains metadata for the executed jobs in H2Hadoop?

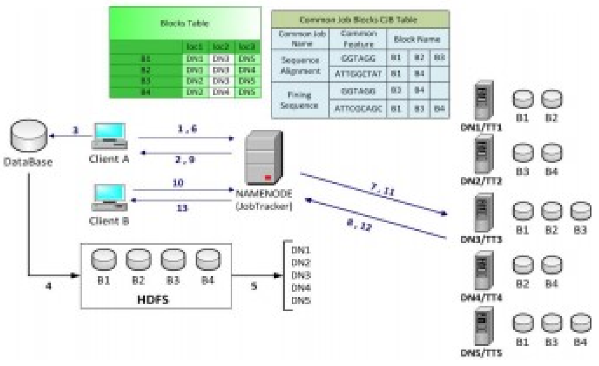


Figure 1. H2Hadoop MapReduce Workflow

MapReduce workflow in H2Hadoop has been explained in figure 4 as follows:

Step 1: Client " A" sends a request to Name Node. The request includes the need to copy the data files to Data Nodes.

Step 2: Name Node replays with the IP address of Data Nodes. In the above diagram Name Node replies with the IP address of five nodes (DN1 to DN5).

Step 3: Client "A" accesses the raw data for manipulation in Hadoop.

Step 4: Client "A" formats the raw data into HDFS format and divides blocks based on the data size. In the above example the blocks B1to B4 are distributed among the Data Nodes.

Step 5: Client "A" sends the three copies of each data block to different Data Nodes.

Step 6: In this step, client "A" sends a MapReduce job (job1) to the JobTracker daemon with the source data file name(s).

Step 7: JobTracker sends the tasks to all TaskTrackers holding the blocks of the data.

Step 8: Each Task Tracker executes a specific task on each block and sends the results back to the JobTracker.

Step 9: JobTracker sends the result to Client "A". In this step, Name Node keeps the names of the blocks that produced the results in the local lookup table (CJBT) by the Common Job Name (Job1) that has common feature as explained above.

Step 10: Client "B" sends a new MapReduce job "Job2" to the JobTracker with the same common job name and same common feature or super-sequence of "Job1".

Step 11: JobTracker sends "job2" to TaskTrackers who hold the blocks, which have the first result of the MapReduce "Job1" (DN2, DN4, DN5). In this step, the JobTracker starts with checking the CJBT first to find if it is a new job which has the same common name and common features of any previous ones or not – In this case yes. Then the JobTracker sends "Job2" only to TT2, TT4 and TT5. We may assume here that the lookup table will be updated with more details OR just remain as is because every time we have a new job that may carry the same name of "Job1".

Step 12: TaskTrackers execute the tasks and send the results back to the JobTracker.

Step 13: JobTracker sends the final result to Client "B".

The workflow that is shown above explains the normal flow steps of the H2Hadoop MapReduce framework. In addition, there should be a training phase before starting the process of MapReduce to have some metadata in the CJBT to receive the benefits of the new architecture. From the flowchart that is explained in Figure 5, we can see that there are two more conditions in H2Hadoop when compared with native Hadoop that perform with a delay in job processing. However, if we have a relationship between jobs, H2Hadoop performance will be better than the native Hadoop. The above-mentioned delay in H2Hadoop ultimately causes a short delay in time.

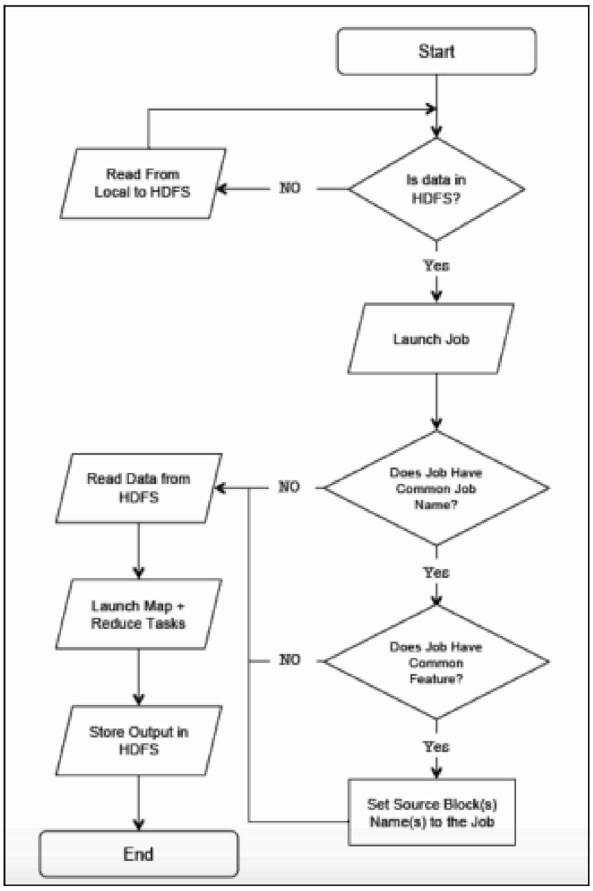

Start
Is data in HDFS?
NO
Read From Local to HDFS
Yes
Launch Job
Does Job Have Common Job Name?
NO
Read Data from HDFS
Yes
Launch Map + Reduce Tasks
Does Job Have Common Feature?
NO
Store Output in HDFS
Yes
Set Source Block(s) Name(s) to the Job
End

Figure 2. H2Hadoop MapReduce Workflow Flowchart

In H2Hadoop, after launching a job there is a condition that tests the name of the job. If the job uses a CJN, which means this job is commonly used and there might be a relationship between this job and others. Otherwise, if the name of the job is not common, it skips the second condition and reads the whole data from the HDFS and completes the execution. If the name of the job is common, which means the first condition is "Yes", it will check the second condition, which tests the common feature of the job. If the feature of the new job is common with any previous job, the new job reads the specific data blocks from the HDFS and sets them as source data files, not the whole data block. Then the new job will be executed normally. Under these two conditions, H2Hadoop reduces the size of the data that is being read by the new job. Consequently, this improves on the Hadoop performance for jobs that are working on similar data files.

II. CONCLUSION

In this work we have a tendency to gift increased Hadoop framework that permits a Name Node to spot the blocks within the cluster wherever sure info is hold on. We mentioned the projected progress in the proposed project and compared the expected performance of proposed system to native system. In this project, we have a tendency to browse less information, so we have some Hadoop factors like variety of browse operations, that area unit reduced by the quantity of Data Nodes carrying the supply information blocks, which is identified before causation employment to Task Tracker. The maximum variety of knowledge blocks that the Task Tracker can assign to the work is adequate the quantity of blocks that carries the supply information associated with a selected common job.

III. REFERENCES

[1]. Ming, M., G. Jing, and C. Jun-jie. Blast-Parallel: The parallelizing implementation of sequence alignment algorithms based on Hadoop platform. in Biomedical Engineering and Informatics (BMEI), 2013 6th International Conference on. 2013.

[2]. Schatz, M.C., B. Langmead, and S.L. Salzberg, Cloud computing and the DNA data race. Nature biotechnology, 2010. 28(7): p. 691.

[3]. Schadt, E.E., et al., Computational solutions to large-scale data management and analysis. Nature Reviews Genetics, 2010. 11(9): p. 647-657.

[4]. Farrahi, K. and D. Gatica-Perez, A probabilistic approach to mining mobile phone data sequences. Personal Ubiquitous Comput., 2014. 18(1): p. 223-238.

[5]. Marx, V., Biology: The big challenges of big data. Nature, 2013. 498(7453): p. 255-260.

[6]. Lohr, S., The age of big data. New York Times, 2012. 11.

[7]. Changqing, J., et al. Big Data Processing in Cloud Computing Environments. in Pervasive

Systems, Algorithms and Networks (ISPAN), 2012 12th International Symposium on. 2012.

[8]. Chen, M., S. Mao, and Y. Liu, Big Data: A Survey. Mobile Networks and Applications, 2014. 19(2): p. 171-209.

[9]. Jagadish, H., et al., Big data and its technical challenges. Communications of the ACM, 2014. 57(7): p. 86-94. 10. White, T., Hadoop: The definitive guide. 2012: "O'Reilly Media, Inc.".

[10]. Patel, A.B., M. Birla, and U. Nair. Addressing big data problem using Hadoop and Map Reduce. in Engineering (NUiCONE), 2012 Nirma University International Conference on. 2012.

[11]. Hammoud, M. and M.F. Sakr. Locality-Aware Reduce Task Scheduling for MapReduce. in Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. 2011. 12Dean, J. and S. Ghemawat, MapReduce: simplified data processing on large clusters. Communications of the ACM, 2008. 51(1): p. 107-113.

[12]. Xiaowen Feng, Hai Jin, Ran Zheng, Zhiyuan Shao, Lei Zhu," Implementing Smith-Waterman Algorithm with Two-dimensional Cache on GPUs " Second International Conference on Cloud and Green Computing, 2012.

[13]. Novan Zulkarnain and Muhammad Anshari, "Big Data: Concept, Applications, & Challenges", International Conference on Information Management and Technology (ICIMTech),2016.

[14]. J. Ramsingh and V.Bhuvaneswari, "Data Analytic on Diabetic awareness with Hadoop Streaming using Map Reduce in Python", IEEE International Conference on Advances in Computer Applications (ICACA), 2016.

[15]. Ming Meng, Jing Gao*, Jun-jie Chen,"Blast-Parallel: The parallelization implementation of sequence alignment algorithm based on Hadoop platform ", 6th International Conference on Biomedical Engineering and Informatics (BMEI 2013) , 2013.

[16]. Saad Khan Zahid, Laiq Hasan , Asif Ali Khan, Salim Ullah, "A Novel Structire of Smith-Waterman Algorithm for Efficient Sequence alignment", ISBN: 978-1-4799-6376-8/15/$31.00 ©2015 IEEE.

[17]. Miss. Anju Ramesh Ekre," Genome Sequence Alignment tools: a Review", 978-1-4673-9745-2 ©2016 IEEE.

[18]. Rohith K. Menon, Goutham P. Bhat and Michael C. Schatz," Rapid Parallel Genome Indexing with MapReduce", http://www.genome10k.

[19]. Merina Maharjan, "Genome Analysis with MapReduce", ttp://hadoop.apache.org/