Scalable Data Integration using Similar Joins to Merge Large Key Value Pairs

Mohan Kumar A V
Department of Computer Science & Engineering
Don Bosco Institute of Technology
Visveswarya Technological University, Belgaum
Bengaluru, India

Dr. Nanda Kumar A N
Department of Information Science & Engineering
New Horizon College of Engineering,
Visveswarya Technological University, Belgaum
Bengaluru, India

***Abstract* — The Huge amount of text data available should be integrated thereby avoiding the squander of storage space, which can be achieved by similar join technique. In this paper, we propose, similar join technique, which is a based on Map Reduce framework used is uses strings for similar join and supports set based and character based functionalities. Key value pairs are generated using signatures. Merge based algorithm can be used implement, which merge large number of value-key pairs which intern reduces the transmission cost. Here we are using "light-weight" filter units to increase the performance, which mainly trims large number of dissimilar value-key pairs with less transmission cost.**

Keywords* — *Data Integration; Map Reduce; similar Join; Hadoop.

I. INTRODUCTION

Data Integration has become an important process as it combines heterogeneous data and results in meaningful and valuable data. It mainly includes virtual integration and materialized warehousing in multiple dimensions [1], such as volume, velocity, variety and veracity. Similar string joins is a core operation in data integration field, which finds similar pairs of strings from a given collection of strings. Many algorithms have been proposed under string similar join. Some of the algorithms have been used only for specified datasets. Many algorithms could not be decided on which kind of data it can be used. This leads to the development for new scalable string Similar join algorithms.

On a large scale based, Map Reduce is a one of the programming model which is used here to process the large datasets, which are based on string similar join. The Map Reduce function processes data in the distributed servers, parallel, provides great redundancy and most fault-tolerant. A Map Reduce function comprises of map () and reduce () functions, map () function performs the filter and sort operation, whereas reduce () performs collective summary operation. The main disadvantage of this method is it is too costly for large datasets. This issues is addressed in Veronica et al [2], they proposed a prefix filtering method which used filter and verification framework steps.

In the first step, i.e. filter stage some tokens were selected from the strings and set of candidate pairs were generated for common tokens. In the second step, i.e. verification step, candidate pairs were verified to generate the final results. But this method had a biggest limitation with low trimming power. Since single token is of small and short and has a low selective and many dissimilar pairs will have a common token and will not be trimmed. To overcome this limitation, a new framework based on Map Reduce called similar Join method is implemented for scalable string similarity joins. It usually incorporates filter frame work and verification framework. In the filter stage, signatures for each string were generated and if two strings are similar, only then they share common signatures. This property is used for the generation of candidate pairs. In the verification stage, candidate pairs will be verified to generate the final result [4, 9].

The Map Reduce function considers strings as value to generate key – value pairs, and signatures as keys. These key value pairs will be used to compute the candidate pairs which share same key. This approach will generate a large number of key value pairs which leads to increase in the transmission cost. To avoid this, merging value-key pairs will reduce the number of value-key pairs without decreasing the trimming power. At last, to increase the performance, "light-weight" filter units are used in key value to trim a large number of dissimilar pairs without increasing the transmission cost.

II. PRELIMINARY

A. Problem Definition

Similar or similarity function is used to find the similar strings pairs from the given set collection of strings. The Similar between the two strings are evaluated by using Similar functions, whereas character based similarity functions and set based functions are the other two types of Similar functions, many algorithms have been proposed to manage Similar joins such as which generates signatures for each string if two or more strings are similar and the signature must be of same [12]. Many filtering techniques were also proposed such as, count filtering [13, 18, 19], length filtering [13, 16], position filtering [7, 20], and prefix filtering [17], content filtering [20]. PartEnum [15], Pass Join [16], FastSS [22] and Triejoin [23].

B. Related work

Character Based Similarity functions checks the number of character operations required to transform one string to another. Distance edit is a type of character similarity function. The Edit operations which can be performed on the string are insertion of a new characters, deletion of a characters and replacing a character by another string. For example, if a="John" and b="John", the ED (a, b) =2, because the string can be obtained by interchanging the 2 characters.

Set-Based similarity functions consider each word as a string and similarity is found between each strings. The

different types of set-based similarity functions are Jacquard Similarity, Dice Similarity and Cosine Similarity. They are defined as below.

$$JAC(a, b) = |a \cap b| / |a \cup b|$$

$$COS(a, b) = |a \cap b| / \sqrt{(|a| . |b|)}$$

$$DICE(a, b) = 2|a \cap b| / |a| + |b|$$

Map Reduce is the heart of Hadoop. It is a simple Programming Model which is designed to process and generate huge data sets [5]. It is reliable scalable and does the parallel processing of large data in a distributed environment [8]. The Map Reduce algorithm consists of two main tasks. Map and Reduce tasks to suitable servers in the cluster. After the task is finished, the cluster collets and reduces the data into proper results and sends it back to Hadoop server.

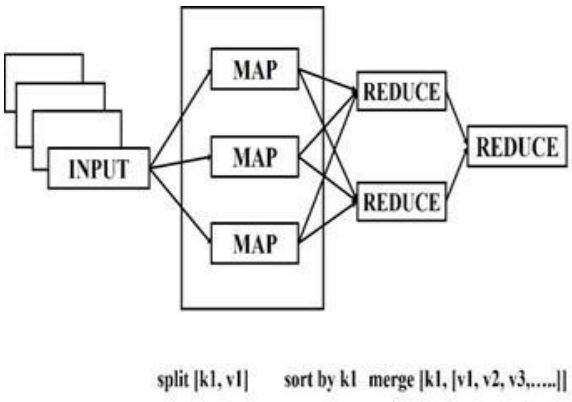

Fig 1: Map Reduce Architecture.

Fig 1, explain the Map Reduce architecture, the input data is split into proper number of sizes and are assigned to map a function. Map function generates intermediate <value-key> pairs of the input data. The intermediate values will be sent to the desired reduce node and the values will be sorted according to keys, then the values will be merged to obtain the final result. V-SMART [6] joins is another scalable Map Reduce based framework which is applicable to sets, vectors and multi sets. It also include two stage algorithm process, where in first step it computes and joins the intermediate results and in second step, it finds the similarity for all the candidate key pairs. Another important issue while generating the candidate pairs is duplicate records.

To address this issue, i.e., to find the duplicate records efficiently, many algorithms were proposed such as, prefix filtering method, positional filtering based method, suffix filtering as mentioned in [7, 11]. Fuzzy Joins the fuzzy joins used two main approaches such as approximate matching and exact matching earlier. Recently fuzzy joins uses Map Reduce [2, 13] to locate the similar sets like by using prefix-based method [12]. Positional and suffix filtering techniques [7] and then parallelizes the techniques. In [3], the methods used for returning the correct output are exact matching techniques.

III. SIMILAR JOIN FRAMEWORK

Similar Join is a Map Reduce framework for string similar joins supports character similar functions and set similar functions. The work flow of similar join algorithm can be explained in three steps: Signature generation stage, filter stage and verification stage.

A. Signature generation

The set-based similarity function generates the signature for each string based on the string length and character-based similarity function generates the signature based on edit distance threshold. Since both the functions use different methods to generate signature, a new method which can generate signatures for both set-similar function and character base similar function are used. The signature generation in similar join algorithm can be done using two methods. Position known method and multi match conscious method. Both the can be used together as hybrid method to generate the signatures. These methods will reduce the number of signature generation significantly and avoids false negatives.

B. Filter stage

Filter stage consists of Map phase first and Reduce phase second. The candidate pairs are generated using the methods explained in signature generation stage. In Map phase, signatures are considered as keys and strings as values. Since two same similar strings have same key, they are scuffled to same reduce task. As well, strings are replaced by string ids which reduce transmission cost. In reduce phase, value key pairs are considered as input which consists of signature and strings list containing the signature. Then it divides the list into two groups for <aid> and <bid>.

C. Verification stage

The Verification stage consists of two stages as seen in Fig. 2. Both the stages again performs Map and Reduce functions in which it eliminates the duplicates candidate pairs which were generated due to two strings sharing multiple signature. The string ids will be replaced by original strings to verify the candidate pairs.

IV. MERGE-BASED ALGORITHMS

The Similar join algorithm generates large number signatures based on value-key pairs. The main goal of the merge-based algorithm to reduce the number of value-key pairs. This method reduces the trimming power because the sub string and the segment may be matching at different start positions or with different lengths. In both the cases, the false positives will get generated. Hence the same pruning power should be retained as similar join algorithm by checking the start position of the substring and length of the string are within the bounds. In order to efficiently merge the original value-key pairs to obtain the new value-key pairs, read the string once to generate all value-key pairs and in reduce phase split the input value list into tow lists to generate the output.

V. LIGHT-WEIGHT FILTERS UNITS

The transmission cost and processing cost in the verification stage is high to generate the candidate pairs. Hence light-weight filter units are used usually to reduce the number of candidate pairs. In Map phase stage of filter, attach the original string for values field in each <value, key> pairs.

In Reduce phase, the similarity for the candidate pairs are calculated and the < value, key > pairs whose similarity observed is less than the threshold present will be deleted.

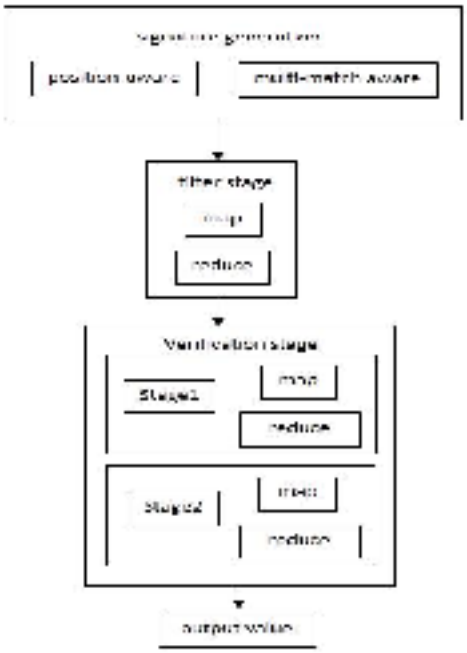

Fig 2: Similar Join Flow Chart.

This method will minimizes the transmission cost of candidate pairs, but found increased transmission cost for of original strings. To address this, the original strings are replaced by light-weight filter units. Also these filter units can be used to trim dis similar pairs.

VI. EXPERIMENATL RESULTS

The Similar join algorithm is tested on twitter data sets. The basic Map Reduce algorithm and similar join algorithm are compared for the results. The basic Map Reduce and Similar join algorithms are implemented on Hadoop on a single node.

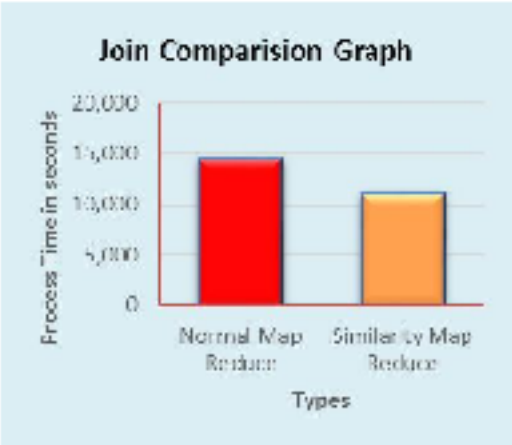

Fig 3: Similar Join Comparison Graph.

The node is installed with 64-bit Ubuntu Server 14.04, Java 1.7 and above and Hadoop 2.7.1. Fig 3 shows the processing time for both basic Map Reduce and similar join algorithms. From the graph we can analyze that the similar join algorithm finishes the processing of data much faster that basis Map Reduce algorithms.

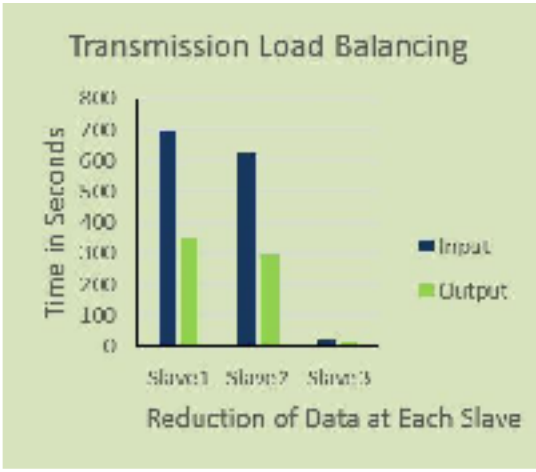

Fig 4: Reduction of data at each slave, Join for static result.

Fig. 4 shows the graph for Join operation of static data, which uses the basic Map Reduce function to reduce the data for three different slaves.

VII. CONCLUSION

Similar Join algorithm for strings similar join is based on Map Reduce which is implemented by incorporating merge algorithm and used the light weight filters for character based similar functions and set similar functions. By using merge algorithm, the number of value key pairs are reduced to large number extent, without reducing the trimming power. Light weight filter units reduces the number of candidate pairs, which will increases the performance and reduces the transmission cost. The Similar join algorithm is able now to process both short strings and large documents.

REFERENCES

[1] Xing Luna Dong, Divesh Srivastva, Big Data Integration. ICDE Seminar, 2013.
[2] Veronica, M. J. Carey, and C. Li. Efficient parallel set-similarity joins using map reduce. In SIGMOD, pages 495–506, 2010.
[3] N. Afrati, A. D. Sharma, D. Menestrina, A. G. Parameswaran, and J. D. Ullman. Fuzzy joins using map reduce. In ICDE, Pages 498–509, 2012.
[4] D. Deng, Y. Jiang, G. Li, J. Li, and C. Yu. Scalable column concept determination for web tables using large knowledge bases. PVLDB, 6(13):1606–1617, 2013.
[5] Dean and S. Ghemawat: Map reduces: Simplified data processing on large clusters. In OSDI, Pages 137–150, 2004.
[6] Metwally and C. Faloutsos. V-smart-join: A scalable map reduce framework for all-pair similarity joins of multi sets and vectors. PVLDB, 5(8):704–715, 2012.
[7] Xiao, W. Wang, and X. Lin. Ed-join: an efficient algorithm for similarity joins with edit distance constraints. PVLDB, 1(1):933–944, 2008
[8] F. Li, B C OOI, M, T, Ozsu AND S, Wu Distributed Data Management using MAP Reduce ACM Compute Survey 2004
[9] Survey of Scalable String Similarity Joins Khalid F. Alfatmi 1, Archana S. Vaidya2 Department of Computer Engineering, Savitri Bai Phule Pune University, and Maharashtra, India
[10] String Similarity Joins: An Experimental Evaluation Yu Jiang, Guoliang Li, Jianhua Feng, Wen-Syan Li, Department of Computer Science, Tsinghua University, Beijing, China SAP Lab
[11] Mian Want, Tiezheng Nie, Derong Shen, Yue Kou, Ge Yu, Intelligent Similarity Join for big data Integration10th Web Information System and Application conference 383-388, 2013.

[12] Ranieri baraglia, Gianmarco De Francisci Morales, Claudio Lucchese Document Similarity Self-Join with Map Reduce, in ICDM '10.
[13] L. Gravano, P. G Iperiotis H. V. Jagadish, N. Koudas, S. Muthukrishna and D. Srivastava, Approximate String Joins in a database (almost) for free. In VLDB, pages 491-500, 2001.
[14] R. J. Bayardo, Y. Ma and R. Srikant Scaling up all Pairs similarity search. In WWW, Pages 131-140, 2007.
[15] A. Arasu, V. Ganti, and R. Kaushi, Efficient exact set similarity joins in VLDB, pages 918-929, 2006.
[16] G. Li. D. Deng, wang and J. Feng. Pass Join: A partition-based Method for similarity joins. PVLDB, 5(3):253-624, 2011.
[17] S. Chaudhuri, V. Ganti and R. Kaushik: A primitive operator for similarity joins in data cleaning. In ICDE, Page 5, 2006.
[18] C. Li, J. Lu and Y. Lu. Efficient merging and filtering algorithm for appropriate string search. In ICDE, Pages 257-266, 2008.
[19] S. Sarawagi and A. Kirpal, Efficient set joins on similarity predictions. In SIGMOD Conference, pages 743-754, 2004.
[20] C. Xiao, W. Wang and X. Lin, Ed-join: an efficient algorithm for similarity joins with edit distance constraints. PVLDB, 1(1): 933-944, 2008.
[21] W. Wang, J. Qin, C. Xiao, X. Lin and H.T. Shen, Venhunckjoin: An efficient algorithm for edit distance similarity joins. IEEE Trans, Knowl. Data Eng. 25(8):1916-1926, 2013.
[22] B. S. T. Bocek, E. Hunt Fast Similarity Search in large dictionaries Technical, Report ifi-2007, 02, Department of Informatics, University of Zurich, April 2007. http://fastss.csg.uzh.ch/
[23] J. Wang G. Li and J. Fang Can we beat the prefix filtering? An adaptive framework for similarity joins and search. In SIGMOD Conference, pages 85-96, 2012.
[24] J. Qin, W. Wang, Y. Lu. C, Xiao and X. Lin: Efficient exact edit similarity query processing with the asymmetric signature scheme. In SIGMOD Conference, Pages 1033-144, 2011.