ISSN No:-2456 –2165

Removing Similar Data Chunk Using Map Reducing Algorithm on Cloud Storage

Fairlin Jenefa. A , Prof. Shajini. N
Ponjesly College of Engineering , Nagercoil, Kanyakumari Dist, Tamil Nadu

Abstract:-Big sensing data is prevalent in both industry and sciatic research applications where the data is generated with high volume and velocity. Current big sensing data processing on Cloud have adopted some data compression techniques. However, due to the high volume and velocity of big sensing data, traditional data compression techniques lack sufficient efficiency and scalability for data processing. Based on septic on-Cloud data compression requirements, a scalable data compression approach is proposed to calculate the similarity among the partitioned data chunks. Instead of compressing basic data units, the compression will be conducted over partitioned data chunks. To restore original data sets, some restoration functions and predictions will be designed. Map Reduce is used for algorithm implementation to achieve extra scalability on Cloud. The proposed scalable compression approach based on data chunk similarity can significantly improve data compression efficiency with affordable data accuracy.

Keywords—Cloud Computing; Big Sensing Data;
Map Re-Duce Algorithm for Big Data; Compression Techniques.

I. INTRODUCTION

The big sensing data is large in volume and it comes from the different sensing systems. This data is very large and it requires processing before storing it on the cloud. Therefore, this purpose that data should be compressed first and then it should be stored on the cloud. There are different sources of the big sensing data such as camera, video, satellite meteorology, traffic monitoring, complex physics simulations [11]. Hence big data processing is a big fundamental challenge for the modern society.

Cloud is a promising platform for big data processing with its powerful computational capability, storage, resource reuse and low cost processing the big sensing data is still costly in terms of space and time. For the reduction of the time and space cost of big data different techniques should be needed.

If some recursive algorithms are used to process the big sensing data it can produce many problems such as memory bottlenecks, deadlocks on data accessing chunks. Here the Jaccard similarity algorithm is used. base paper the Cosine similarity algorithm was used. In this paper we are comparing the both algorithms on the factor accuracy of the data. The accuracy and the space cost parameters are directly proportional to each other i.e. as the accuracy of the data increases the more space is required to store that data. The similarity algorithm Jaccard is used to find the data similarity more accurately than Cosine similarity algorithm. In results we will compare both the algorithms i.e. Jaccard algorithm and cosine similarity algorithms on the basis of the accuracy of the data preserved i.e. how accurately they finds the similarity of the data. As compared to the Cosine similarity algorithm the Jaccard algorithm is more advantageous in some measures as given in [5]. After finding the similarity by Jaccard algorithm another algorithm is applied to the data chunks is known as Map Reduce algorithm. The Map Reduce algorithm is used is for the compression of the similar data chunks.

In this paper, the similarity data chunks are generated at first. Then the similarity algorithm is applied on these data analysis which shows significant speed up.

II. REVIEW OF LITERATURE

[1] shows that the cloud can also be used for the storage. In [1]cloud is defined. It also provides architecture of the cloud with market-oriented resource allocation by using technologies such as virtual machines (VMs).It also describes the market based resource management strategies that covers both customer-driven service management and sustain the service level agreement.

The paper [2], describes the general algorithmic design technique in the Map Reduce framework called filtering. The main purpose behind the filtering is to reduce the size of input in the distributed fashion. By filtering resulting much smaller, problem instance can be solved on single machine. Using [2] the different approaches new algorithms in the Map Reduce framework for a variety of fundamental graph problems for sufficiently dense graphs. In this algorithm the amount of memory available on the machines allow us to show tradeoff between the memory which is available and the number of Map Reduce rounds. This implements the

maximal matching algorithm that lies at the core of the In [3], introduces the Map Reduce framework based on the Hadoop. This shows the efficient Map Reduce algorithm and the present state of art in the Map Reduce algorithm for the data mining machine learning and similarity joins.

In paper[4], introduces the two fundamental technologies is the distributed data store and another is complex event processing and shows the work flow description for distributed data processing.

In Paper[5], Jaccard coefficient is used as an information similarity measure. In addition of the similarity measure Jaccard has the advantage which protects the privacy of the data. They proposed the SJCM protocol (Secure computation of the Jaccard Coefficient for Multisets) using the existing dot product method.

Paper [6], shows the definition of the anomaly detection and the big data. The anomaly detection is based on the uncompressed data due to storage burden and the inadequacy of privacy protection and compressive sensing theory introduced and used in the anomaly detection algorithm. This anomaly detection technique used for the through-wall human detection to demonstrate the effectiveness.

Paper [7] consists of following folds:1)scientific communities which are small to medium scale utilize the elastic resources on the public cloud site while maintaining their flexible system control. 2)Lightweight service management for making management tasks simple and service heterogeneous workloads.

Paper[8]gives the approaches and mechanisms of deploying intensive data applications which are gaining scalability, consistency, economical processing of large scale data on the cloud and also highlights some characteristics of the best candidate classes.

Paper[9], describes a work-flow manager which is developed and deployed at Yahoo Calles Nova. This manager pushes the continuously arriving data through graphs of programs which are executing on Hardtop clusters. The programs are known as pig which is structure flow language.

Paper[10], describes the history record based service optimization method. In [11], the services are selected based on the history records of the services. This Hire some-II method based on the history records therefore the privacy of the cloud is protected.

In [12],the framework called Early accurate result library(EARL) is described. This framework chooses the appropriate sample size for achieving the desired error. The error estimation based on technique called bootstrapping.

III. SYSTEM ARCHITECTURE / SYSTEM OVERVIEW

A. System Description

In proposed work, main algorithm is used top process big sensing data. So, some features of big sensing data will be studied and analyzed. To carry out compression, the similarity between two different data chunks should be defined. So, how to define and model the similarity between data chunks is a primary requirement for data compression. After the definition for the above similarity model for data chunks, how to generate those standard data chunks for future data compression is also a critical technique which we designed. A novel compression algorithm is developed and designed based on our similarity model and standard data chunk generation.

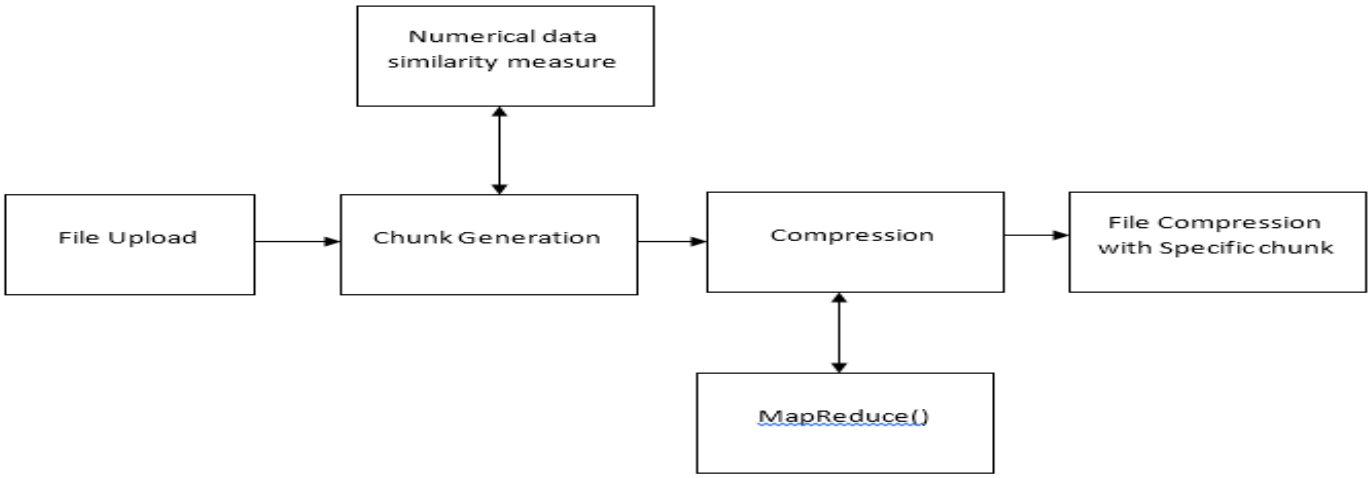

Figure. 1. System Architecture

a). Data Chunk Similarity and Compression

- Data chunk based compression recognizes complex data partitions and patterns during the compression process.
- Systems are often compressed by means of Deduplication techniques, that partition the input text into chunks and store recurring chunks only once.
- The similarity is described with a cos u between two vectors and fraction between two matrix norms of modulus of X vector and Y vector.
  Simn1$(\vec{x},\vec{y})$=cos θ

$$\cos\theta = \frac{X_1y_1 + X_2y_2 + \cdots + x_ny_{n_2} + \cdots + x_ny_n}{\|\vec{x}\| \times \|\vec{y}\|}$$

$$\cos\theta = \frac{x_1y_1 + x_2y_2 + \cdots + x_ny_n}{\sqrt[2]{x_1^2 + x_2^2 + \cdots + x_n^2}\, x\, \sqrt[2]{y_1^2 + y_2^2 + \cdots + y_n^2}}$$

measure similarity between two vectors x and y, a popular similarity function is the inner product including the cosine similarity, Pearson correlations, and OLS coefficients. The inner product is unbounded. One way to make it bounded between -1 and 1 is to divide by the vectors' norms. It is called the cosine similarity.

- cosine similarity can measure the big data chunk similarity more accurately under our big sensing data feature assumption.

b). Data Chunk Generation and Formation

- The data will not be compressed by encoding or data prediction one by one. It is similar to high frequent element compression by Map Reduce algorithm. Data chunk based compression recognizes the complex data partitions and patterns during the compression process.
- In data chunk generation, the recursive process will increase the size of selected standard data chunks step by step.
- The formation process of data chunks will be offered by training initial data stream.
- At first, Compare the largest data chunk to find a similarity data chunk.
- The size of the standard data chunk is controlled by selecting the parameter 'r' which determines when to terminate the recursive process of generating new data chunks;
- where r is time round.
- the generated standard data chunks after initial selection have different size to each other. In this data chunk generation, the recursive process will increase the size of selected standard data chunks step by step.
- If the terminating condition is not reached, the algorithm will continuously find new standard chunks with bigger size compared to any standard chunk which has already been selected.
- When the terminating condition is reached according to the given 'r', the final selected standard data chunk is always the largest one in terms of size.

c). *Data Chunk Based Big Data Compression*

- Develop a new data compression technique which recursively compresses in-coming data from big data set.
- Data chunk based compression, data sets should be compression block by block. For big graph data and lots of network data, the topology and structure information has big influence for data processing and it should not be ignored.
- With the formed S' and a recursive process, the big sensing data stream will be compressed for both space and time efficiency; Where S' is the first initial standard data chunk set.

d). Map Reduce

- Map Reduce is a framework for processing parallelizable and scalable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogeneous hardware).

- The Map Reduce algorithm contains two important tasks, namely Map and Reduce.
  - The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).
  - The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples.
- The reduce task is always performed after the map.

e). Data Compression Module

- Data compression has important application in the field of file storage and distributed systems. It helps in reducing redundancy in stored or communicated data.
- Reducing the size of a file to its half is equivalent to doubling the storage medium capacity. On the downside, compressed data needs to be decompressed in order to view the data and this extra processing may prove detrimental to some applications.
- Data Compression is the process of encoding data so that it takes less storage space or less transmission time that it would if it were not compressed. Compression is possible because most of the real world data is very redundant.

Data Compression is a technique that reduces the size of data by removing unnecessary information and duplicity. Data compression is the art of reducing the number of bits needed to store or transmit data.

B. Purpose

The proposal of novel technique based on data chunk partitioning for effectively processing big data, mainly streaming big sensing data in Cloud. Map Reduce is used for algorithm implementation to achieve extra scalability on Cloud. A novel compression algorithm is developed and designed based on our similarity model and standard data chunk generation.

a). Map Reducer Algorithm

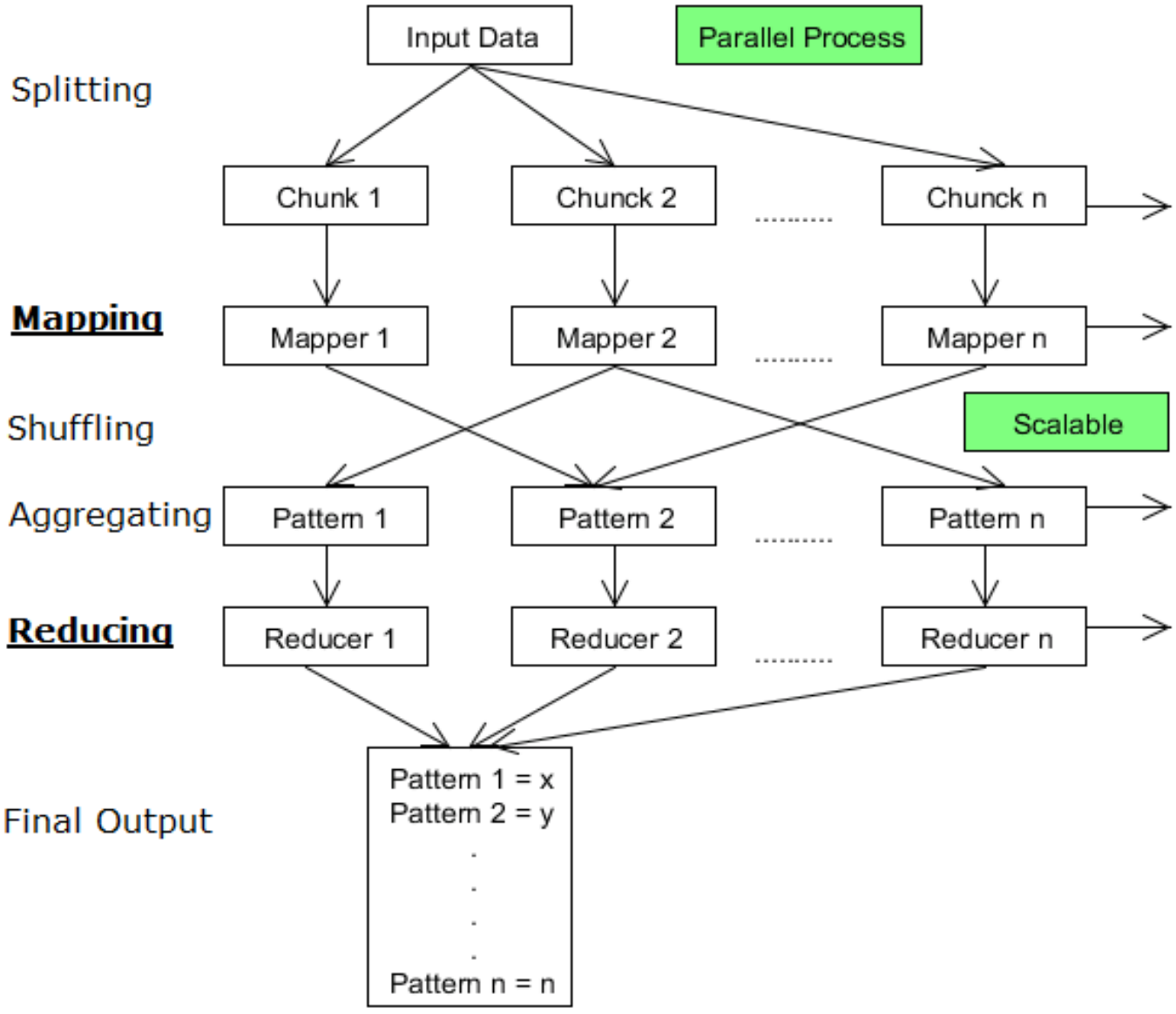

Figure 2: Flow Chart for Map Reduce Algorithm

- Map Reduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster.
- Map Reduce is as a 5-step parallel and distributed computation:
- Prepare the Map() input – the "Map Reduce system" designates Map processors, assigns the input key value K1 that each processor would work on, and provides that processor with all the input data associated with that key value.
- Run the user-provided Map() code – Map() is run exactly once for each K1 key value, generating output organized by key values K2.
- "Shuffle" the Map output to the Reduce processors – the Map Reduce system designates Reduce processors, assigns the K2 key value each processor should work on, and provides that processor with all the Map-generated data associated with that key value.
- Run the user-provided Reduce() code – Reduce() is run exactly once for each K2 key value produced by the Map step.
- Produce the final output – the Map Reduce system collects all the Reduce output, and sorts it by K2 to produce the final outcome.

IV. RESULTS DISCUSSION

A. Space and Time Saving from Compression

The overall compression effectiveness of our proposed data compression is demonstrated. The X axis stands for the incoming big sensing data size for testing. The Y axis stands for the compression achieved by deploying our compression algorithm. There are two important testing endings should be indicated. With the increase of R from 10 to 90 rounds, the compression ratio increases whatever the testing data size is from 1 to 10 terabytes. In other words, the larger R brings more compression ratio and performance gains to our compression algorithm. It can be noticed that with the increase of testing data size, the compression ratio decreases whatever the value of R is. This result indicates that with more data gathered for testing, more heterogeneous data blocks could be found in meteorological big sensing data set. In other words, more new data blocks which are not compressible with the standard data chunks set could be detected.

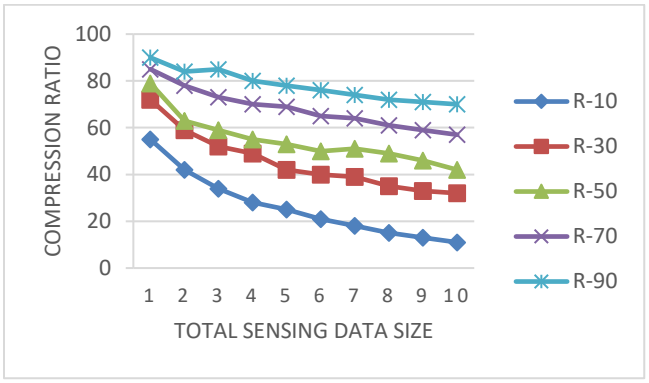

Fig 3: Compression Ratio for Different ‘R’.

B. Data Accuracy Analysis

First the accuracy dentition is briery described according to the work based on measuring the similarity between two vectors, one from real big data graph G and the other from G’ altered data as the service provided by Cloud. There are two vectors at a certain time stamps.

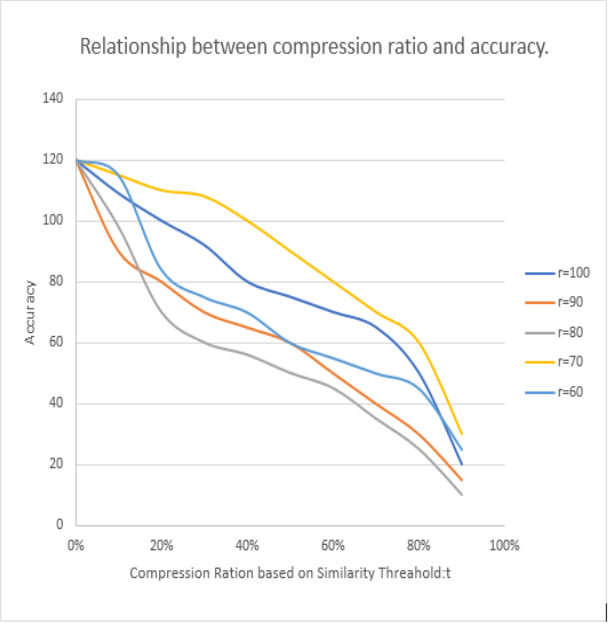

Fig. 4 Relationship Between Compression Ratio and Accuracy.

To describe the similarity between two nodes, correlation coefficient model is used as shown in. Suppose X from G and Y from G’ are two vectors. With Correlation Coefficient method, we can calculate the similarity between them by formula.

$$\mathrm{sim}(x,y) = \mathrm{r}(x,y) = \frac{cov(x,y)}{\sqrt{cov(x,y).(y,y}}.$$

From we can find that this similarity resembles to the "cos" similarity computation. sim (X, Y) has a data range The final accuracy for "Accuracy" between two points within a cluster

$$\text{Accuracy} = \sum_{t=0}^{T} \left(\frac{\left| \sum_{i=1}^{n} (it - X_{t)(Y_{it} - Y_t)} \right|}{\sqrt{\sum_{i=1}^{n} (X_{it} - X_t)^2 \, (Y_{it} - Y_t)^2}} . \right) / T * 100\%$$

Suppose that in graph data set G(V, E), there are total S edges (with cluster-head structure, edge explosion is avoided) and each edge is indexic with s from . We can calculate the Average Accuracy of the Cloud Computed Data set 'G' against the original 'G'. This Average Accuracy is used in formula (25) to demonstrate our experiment results in Fig. 10

$$\text{Average} - \text{Accuracy} = \sum_{s=1}^{s} (Accuracy)\, s.$$

With the dentition of above data accuracy, the data accuracy test is designed and conducted. The testing results are demonstrated in Fig. 10. Specifically, we use as the parameter R from 10 to 100 rounds for conducting accuracy test. The increase of compression ratio from 0 to 80 percent, the data accuracy decreases dramatically. That higher the R is, better the data accuracy can be achieved. The reason is that a larger R means more standard data chunks, hence a more refined similarity comparison to guarantee better data accuracy.

C. Experimental Setup

For experiments big data storage on Cloud platform, to reduce the data size also means the time saving for navigating and decompressing those data units. Main designing target of our compression algorithm based on data chunks similarity is to reduce the data size and volume; we also consider the data quality and fidelity loss after deploying our proposed compression and decompression.

V. CONCLUSION

The proposed a novel scalable data compression based on similarity calculation among the partitioned data chunks with Cloud computing. A similarity model was developed to generate the standard data chunks for compressing big data sets. Instead of compression over basic data units, the compression was conducted over partitioned data chunks. The MapReduce programming model was adopted for the algorithms implementation to achieve some extra scalability on Cloud. With the real meteorological big sensing data experiments on our U-Cloud platform, it was demonstrated that our proposed scalable compression based on data chunk similarity significantly improved data compression performance gains with affordable data accuracy loss. The significant compression ratio brought dramatic space and time cost savings.

VI. FUTURE WORK

With the popularity of Spark and its specialty in processing streaming big data set, in future we will explore the way to implement our compression algorithm based on data chunks similarity with Spark for better data processing achievements.

REFERENCES

[1]. S. Tsuchiya, Y. Sakamoto, Y. Tsuchimoto, and V. Lee, "Big data processing in cloud environments," FUJITSU Sci. Technol. J., vol. 48, no. 2, pp. 159–168, 2012.
[2]. "Big data: Science in the petabyte era: Community cleverness Required" Nature, vol. 455, no. 7209, p. 1, 2008.
[3]. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50–58, 2010.
[4]. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Gener. Comput. Syst., vol. 25, no. 6, pp. 599–616, 2009.
[5]. L. Wang, J. Zhan, W. Shi, and Y. Liang, "In cloud, can scientific communities benefit from the economies of scale?," IEEE Trans. Parallel Distrib. Syst. vol. 23, no. 2, pp. 296–303, Feb. 2012.
[6]. S. Sakr, A. Liu, D. Batista, and M. Alomari, "A survey of large scale data management approaches in cloud environments," IEEE Commun. Surveys Tuts., vol. 13, no. 3, pp. 311–336, Jul.-Sep. 2011.
[7]. B. Li, E. Mazur, Y. Diao, A. McGregor, and P. Shenoy, "A platform for scalable one-pass analytics using mapreduce," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2011, pp. 985–996.
[8]. R. Kienzler, R. Bruggmann, A. Ranganathan, and N. Tatbul, "Stream as you go: The case for incremental data access and processing in the cloud," in Proc. IEEE ICDE Int. Workshop Data Manage. Cloud, 2012, pp. 159–166.
[9]. C. Olston, G. Chiou, L. Chitnis, F. Liu, Y. Han, M. Larsson, A. Neumann, V. B. N. Rao, V. Sankarasubramanian, S. Seth, C. Tian, T. ZiCornell, and X. Wang, "Nova: Continuous pig/Hadoop workflows," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2011, pp. 1081–1090.
[10]. K. H. Lee, Y. J. Lee, H. Choi, Y. D. Chung and B. Moon, "Parallel data processing with mapreduce: A survey," ACM SIGMOD Rec., vol. 40, no. 4, pp. 11–20, 2012.
[11]. X. Zhang, C. Liu, S. Nepal, and J. Chen, "An efficient quasi-identifier index based approach for privacy preservation over incremental data sets on cloud," J. Comput. Syst. Sci., vol. 79, no. 5, pp. 542–555, 2013.
[12]. X. Zhang, C. Liu, S. Nepal, S. Pandey, and J. Chen, "A privacy leakage upper-bound constraint based

approach for cost-effective privacy preserving of intermediate datasets in cloud," IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 6, pp. 1192–1202, Jun. 2013.

[13]. X. Zhang, T. Yang, C. Liu, and J. Chen, "A scalable two-phase top down specialization approach for data anonymization using systems, in map reduce on cloud," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 2, pp. 363–373, Feb. 2014.

[14]. W. Dou, X. Zhang, J. Liu, and J. Chen, "Hire Some-II: Towards privacy aware cross-cloud service composition for big data applications," IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 2, pp. 455–466, Feb. 2015.

[15]. J. Conhen, "Graph twiddling in a map reduce world," IEEE Comput. Sci. Eng., vol. 11, no. 4, pp. 29–41, Jul./Aug. 2009.

[16]. Chi Yang and Jinjun Chen, "A Scalable Data Chunk Similarity BasedCompression Approach for Efficient BigSensing Data Processing on Cloud," IEEE Transactions on knowledge and data Engineering, vol. 29, No. 6, June 2017.