International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

Mapreduce: Simplified Data Processing on Clusters with Privacy Preserving By using Anonymization Techniques

Ashutosh Dixit, Nidhi Tyagi

Abstract*: *Computerized Data from various sources, such as remote sensors, cutting-edge sequencing of bioinformatics and high-performance instruments, are increasing at extremely high speeds. To keep analyzing through results for programming, facilities and measurements, The Researches have to use new procedures and techniques. Google's team started MapReduce programming system which aims to manipulate huge data sets in disseminated frameworks; this design lets software engineers create applications that are extremely valuable to large data processing. The motive of this paper is to explore MapReduce research techniques and to increase current research efforts to improve the execution of MapReduce and its capabilities.

Keywords: Anonymization, Big data, Cloud Computing, MapReduce, Programming Model, Scalability.

I. INTRODUCTION

During the last few years The researcher designed a new procedure that helps us, developers and many others have done hundreds of specific purpose calculations at Google that process a large amount of big data such as crawled files, web-request logs, and so on. For calculating different types of defined data, e.g. updated tables, different portrayals of web site diagram layout, rundowns of the quantity of pages crawled per host. The Organizations are questioning mostly in a given day, and so on. These computations are adroitly guided for the most part. The knowledge information's are relatively huge in these computations, and the computations have to be distributed cross-sectionally over hundreds or thousands of machines to finish in a appropriate amount of time. The program helps with the problems of how to balance the process, how to spread the information and how to highlight disappointments. As above mentioned complexity The researcher has to organize another deliberation that helps us to articulate the straight forwardness of the computations they tries to talk to but hide the disorderly details of parallelization, adaptation to non-critical failure, distribution of information and adjustment of the burden in a library.

Manuscript received on February 10, 2020.
Revised Manuscript received on February 20, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Ashutosh Dixit, Research Scholar, Bhagwant University, Ajmer, Rajasthan
Nidhi Tyagi, Professor, MIET, Meerut

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

The abstraction is designed with the help of Map and Reduces primitives that are present in Lisp and in various other languages. The Researcher discovers that over computations a guide activity has been incorporated into each coherent "record"
in our contribution to requesting the processing of a lot of middle key / value pairs and then applying a reduce function to all of the qualities that have a similar key to each other in order to fit the inferred data. Suitably, the users can apply this functional model particularly, they can use it for Map and Reduce operations that will provide them to parallel large computations in easier ways and use re-execution as the primary mechanism for fault tolerance [1] The main motives of this work are exceptionally basic and incredible interface which gives automatic parallelization and transmission of enormous computations of large scale; it can be easily combined with the use of this interface that achieves high performance on large cluster of commodity.

II. MAPREDUCE BACKGROUND

By Map Reduce Researcher can handle enormous information volume requires data to be disseminated in hundreds or thousands of Computers to complete processing tasks in a very short time. Google's group created MapReduce which operates automatically parallel. It computations; MapReduce was designed to supervise data partition and transfer between computing nodes. In addition, it is very useful for handling and managing node failures. MapReduce provides the very useful platform for the programmers who are connected to big data processing and in designing the software. Programmers have to focus on the problematic situations and with the help of MapReduce they control distributed computation problems, Apache Hadoop is a good example of an open source of MapReduce outside Google [2]. Due to its effective technology, programmers and Reseacher use Hadoop in their research.
For two capacities, software engineers are limited: Map and Reduce. Map work's input needs to be spoken to as a key/value pair; for example, the guide research which conducts a lot of document to sort word considerations takes a document as a key and the document's content as a quality. A lot of middle key / values are given by the guide function. If a word count happens, the transition keys will be single terms and the sum of activities of these words will be the price. Reduce work's contribution is the yield of Map work (the middle of the key/value sets);

it forms these qualities and yields them in records; these papers could be used for other MapReduce cycles.

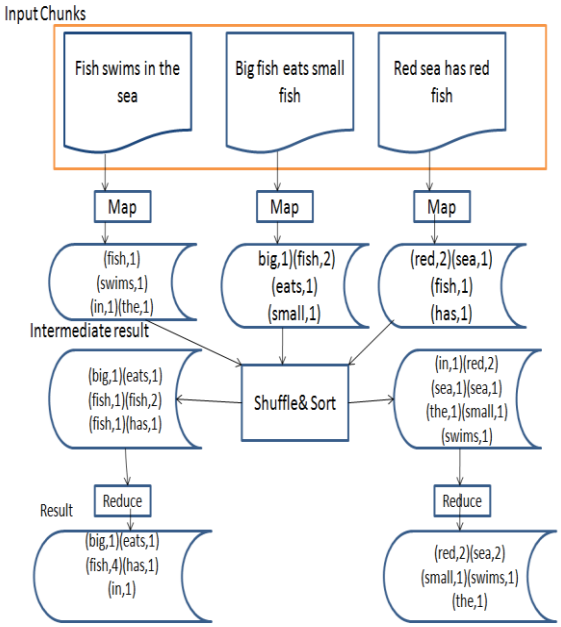

Figure 1: MapReduce process for counting the number of occurrences for each word in a collection of documents

MapReduce perfectly uses network bandwidth by transferring computation to data. Google File System (GFS) manages the data. GFS creates partition of input data into a set of Blocks; the size of the block can be decided by the users. GFS works for repetition of each block. By default, there are three replication GFS sends one replica in the same rack and other replica planning distribution is very useful for the programmer to recover the data in the case of node or rack failures. Here MapReduce collects data locality.

MapReduce works in cluster of nodes; one node works as a master node and other nodes works as workers. As a worker it is the responsibility of worker nodes. Worker nodes fulfill the responsibilities to execute Map and Reduce tasks and master node also fulfill the responsibilities to assign tasks for idle workers. It is the duty of every Map worker to read the content of it's related to split and extracts key/ value pair and sends it to the user for Map functions. The output of this processing is buffered in memory and here it divided into a set of partitions that are equal to number of reducers [31].

Master tells the through workers to read the map workers to information from local disks. These connecting the information or output to the reference are being stored in files. Users can use these files for another MapReduce call, or use them for a particular distributed application.

A. MapReduce Example

If the user's needs to count the number of event for each word which are collected of documents, the map function input is the store of documents, every document represents a record. In this procedure the input key is assigned as a document Id for map function and value is string that presents content of document. The Map function disseminates each file into a sequence of words w1, w2, w3,…,wn. After this key value create the list of pairs. In this procedure every words represents the key and the user always found the value is 1. In this example, the addition is associative and commutative operation; so combiner function can be used to aggregate and it is repeated words that can be occurred in the same document [4].

When Map phase is finished, Shuffle & Sort phase starts. It uses the intermediate data which is generated during Map function, this function works for sorting the data with intermediate data from different nodes, and create the partition of the data into regions then processing is done for reducing tasks. In the end, Reduce function has to use keys and a list of values. For instance, shown word count example that it is mentioned in Fig.1; it uses a fish as key and a list of values {1, 2, and 1}. The last result can be collected into one file that containing every word which is related to its frequency [29].

B. Dealing with Failure

MapReduce is very capable function that can deal with hundreds or thousands of commodity machines. Therefore, it is able to tolerate machine failure. The failure may be appear in master node and it may be in worker nodes. In the matter of master failure all MapReduce function will not be worked, and user can restart after assigning new master node.

The second thing, if user wants to find out worker failure, the master checks all the workers time to time and it also checks the worker status. In a suitable amount of time, if the worker doesn't respond to master ping at this time, the masters notice the failure of worker. In the case of map worker failure, any map tasks either it is in progressive condition or completed by the worker are reset back to its beginning positions and it has to assign for another worker. During in the case of failure in reduce task worker, after it needs to transfer to an idle worker. The result of completed reduce tasks is available in global file system, so it is not essential to re-execute for this completed task. In the matter of, the map tasks, the output is stored in local disks, so the user wants to complete map tasks they have to re-execute in the failure condition.

C. Dealing with Straggler

Straggler is used to a machine that completes map and reduce task. The user can notice that this machine uses long time for such types of task. There are lots of reasons of straggler such as, a machine with bad disk. It usually takes

a long period for completion of its map or reduce task. Due to several reasons, Stragglers appears such as, a machine working with bad disk condition. MapReduce follows general mechanism for reducing the problem of straggler. When the MapReduce task that is near to complete, here master has to work for backup execution. When the primary or backup work execution is completed, the task is showed as completed.

III. PROGRAMMING MODEL

The computation takes a lot of matches between data key/value and creates a lot of key/value sets. The MapReduce library application presents the estimate as two capabilities: Map and Reduce.

Guide, written by the user, takes a knowledge pair and provides a lot of key/value sets in the middle of the road. The MapReduce library bunches all intermediate key connected to a common transfer key I and transfers them on to the job Reduce.

The Reduce function, which is also written by the client, identifies a transitional key I and many virtues for that key. It combines these qualities in order to frame a possibly smaller qualities arrangement. Only zero or one yield esteem is generated by Reduce summons on a regular basis. Using an iterator, the middle values are given to the reduced function of the company. This allows us to deal with qualities arrangements that are too big to even think about memory fitting.

IV. IMPLEMENTATION

The user can do different types of implementations with the help of MapReduce interface. It may be a suitable choice that depends on the environment. For instance, one implementation may be used for a small shared-memory machine, another is useful for a large NUMA multi-processor, and yet another is useful for an even larger collection of networked computers.

This section represents an implementation there are targeted to the distributed environment in most use at Google: Large clusters of commodity PCs connected together with switched Ethernet [6]. In our environment:

(1) Machines are typically dual-processor x86 processors running Linux, with 2-4 GB of memory per machine.

(2) Equipment for the administration of commodity systems is used–either 100 megabits/second or 1 gigabit/ second at machine level on a regular basis, but with an impressive reduction in transmission capacity by and large division.

(3) Machine failures are common activity because a cluster has hundreds or thousands of machines.

(4) Inexpensive IDE provides storage; every machine is directly connected with a disk. A distributed file system has to use replication for providing availability and reliability on the top basic unreliable hardware.

(5) Users submit to the booking system employments. Each activity involves a lot of errands and is mapped to many accessible machines within a group by the scheduler.

V. EXECUTION OVERVIEW

The Map summonses are distributed through different machines by splitting the information data automatically into many sections of M. Different computers may arrange the flow of data in parallel. Lessen summons is distributed by using a dividing power (e.g., hash (key) mod R) to divide the transfer key storage into R bits. The number of allotments (R) and the efficiency of the parceling are calculated by the consumer.

Figure 1 shows the overall progression in our execution of a MapReduce activity. The accompanying succession of activities occurs at the point when the client program calls the MapReduce work (numbered names in Figure 1 correspond to the numbers in the rundown below):

1. In the client program, the MapReduce library first divides the data records into M bits of normally 16 megabytes to 64 megabytes (MB) per piece (controllable by the client using a discretionary parameter). It fires up multiple software duplicates on a group of machines at that point.
2. One of the copies of the program is special i.e. the master. The remaining is workers that are assigned work by the master. Here *M* represents for map tasks and *R* represents for reduce tasks that are assigned. The master catches idle workers and assigns each one a map task or a reduce task.
3. The substance of the related input split is perceived by a laborer who is doled out of a guide task. It parses key/value matches from the knowledge data and sends each pair to the Map function defined by the user. Cradled in memory is the middle of the road key/value sets provided by the Map work.
4. Intermittently, the buffering pairs have to write to local disk, partitioned into *R* regions by the partitioning function. The locations of these buffered pairs on the local disk are sent back to the master,
5. Who is responsible for forwarding these locations to the reduce workers.
6. When a reduce worker focuses on these locations, that is instructed by master, it uses here remote strategy for calls and to read the buffered data from the local disks of the map workers. When a reduce worker read all intermediate data, then it sorts the data by the intermediate keys so that all the occurrence of the same key are grouped together. The sorting process is necessary for various keys map to the same reduce task. If the amount of intermediate data is very large to save in memory, an external sort is used [28].
7. The reduce worker iterates over the sorted intermediate data and for each unique intermediate key en- countered, it passes the key and the corresponding set of intermediate values to the user's *Reduce* function. The output of the *Reduce* function is appended to a final output file for this reduces partition.
8. When all map tasks and reduce tasks have been completed, the master wakes up the user program. At this point, the MapReducecall in the user pro- gram returns back to the user code.

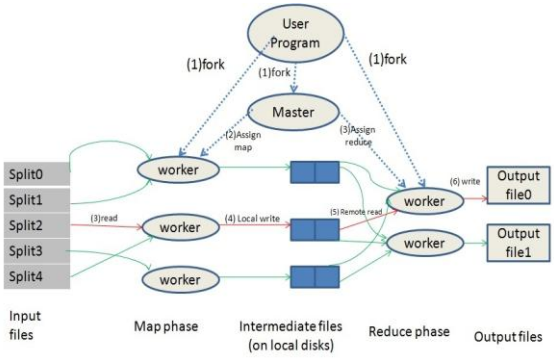

Figure 2: Execution overview

The output of the MapReduce execution was available in the R output files once we successfully completed it (one per reduced function, with file names as indicated by the user).

Most of the time, users are not allowed to merge these R output files into one file–users must transfer these files to another MapReduce request as input, or use them from another distributed program that is ready to handle multi-file input[8].

Semantics in the Presence of Failures

When the client receives delineated decrease administrators are in compliance with the elements of their assessments of data, our distributed use is prepared to produce a similar yield around that time. That's the whole program's non-blaming sequential execution.

To obtain this property, we need to rely on guide nuclear submissions and diminish task yields. In the private transitory documents, each progress task is referenced in its yield. One such record has been produced by a diminished task, and such documents (one for each lessened task) need to be produced by a guide task. The worker needs to make an impression on the ace at the point when a guide task is finished and recalls the names of the R brief documents for the message. If the ace arrives to worry about the guide mission that has been done before, it disregards the message. Else, the names of R documents are put away in a data system of ace.

The lessen expert molecularly transfers the yield file to the last yield documents at the stage when a declining mission stops. In case the same diminish function is conducted on different machines, a similar last yield record has to be done in different ways. We rely on the nuclear rename activity provided by the basic document framework to ensure that only the data created by one execution of the lessen task is contained in the last record framework state.

By far the bulk of our guidance and decrease managers are deterministic, and the manner in which our semantics are equivalent to successive executions for this situation makes it easy for programmers to think about the actions of their software. At a time when the guidance and additionally decreasing managers become non-deterministic, we are offering more unstable and critical semantics. In the hands of non-deterministic executives, the yield of the particular reduction function R1 is proportional to the yield of R1 produced by the successive application of the non-deterministic program. In any case, the yield for an alternative reduction task R2 can be compared with the yield for R2 created by the non-deterministic program's alternate consecutive execution [9].

Find map mission M and lower R1 and R2 undertakings. Let e (Ri) be the execution of Ri that has been submitted (there is actually one such execution). The flimsier semantics emerge on the grounds that e(R1) may have perused the yield of one execution of M and e(R2) may have perused the yield of another execution of M.

VI. DATA PARTITION

The parcel of information is running on the internet. Here a large number of data sets are included. The broad data needs to be shared in small data sets. This gives the arbitrary number for each data set after this. Partitioning is the way to find out which reducer model is going to collect which transitional keys and qualities. -mapper is trained and decides the reducer can get them for the output (key, value) sets. It is essential that they need to lessen together for a key, which is produced in mapper case, created in two separated (key, value) sets. For the reasons of the exhibition, it is also noteworthy that the mappers should be free and capable of autonomously dividing the data they should never have to trade data with each other to decide the parcel for a specific key [10]. It is critical that the target segment is the equivalent for any element, paying no attention to which mapper event it was made. On the off probability of having the primary feline in two separate (heart, esteem) sets, all of them have to be diminished together. It is also important for execution reasons that the mappers have the option to freely parcel data that they should never have to trade data with each other in order to decide on a specific key for the segment.

ALGORITHM: DATA PARTITION MAP & REDUCE.

Input: Capturing Data (ID_m, m), m € D, partition parameter n.
Output: D_i, $1 \le i \le n$.
Map: Generate a random number m and, where $1 \le rand \le n$; emit (m and, m).
Reduce: For every m and, emit (null, list(m)).

Once we partitioned data sets D_i, $1 \le i \le n$, are obtained, we execute MRTDS (D_i, k_I ,AL^0) on these data sets in parallel to evaluate the center of the intermediate anonymization levels AL^*_I ,$1 \le i \le n$.

MERGING

All medium rates of anonymization are translated into one at the next step. The convergence of levels of anonymization is completed by consolidating cuts. Cut_a in AL'_aandCut_b in AL'_bbe two cuts of an attribute.

There exist domain values q_a€Cut_a and q_b€ Cut_b that fullfill one of the three parameters: q_a is identical to q_b, q_a is more general than q_b, or q_a is more specific than q_b. To satisfies that the merged intermediate anonymization level AL_I never violates privacy need, the more general one is selected as the merged one, for instance, q_a will be selected if $q_a$is more general than or identical to q_b. For this scenario of multiple anonymization levels, it can be merged into the same process for the each iteration. According to lemma provides that AL_I still compiles privacy requirements [11].

BIG DATA ANALYTICS SPECIALIZATION

The training in Large Data Analytics is extremely valuable to understudies and will train the understudies to address genuine issues In combination with each of these measurements. For models, the storage of terabytes and even petabytes of data in hundreds of data vaults supporting a huge number of uses is not regular for advanced documents. Maintaining such data vaults requires information in dispersed frameworks of ultra-large scale, innovations in virtualization, distributed computing, unstructured and semi-organized data executives, enhancement strategies dependent on data replication and data movement, as well as cutting-edge data assurance systems. The exponential growth of the amount of data calls for skill in state-of - the-art dynamic data storage approaches combines scalable data processing technologies and developments, manager data streams, and large-scale system analysis, simulation, and extraction [30].

To order to thoroughly investigate these volumes of data from divergent and incompatible commands, data specialists should use propelled data integration processes and business intelligence infrastructure, publicly supporting advances, large-scale software mixes, serious data processing and semantic data management [12].

ALGORITHM: DATA SPECIALIZATION MAP & REDUCE.

Input: Data record (ID_m, m), n € D.; Anonymization level AL*.

Output: Anonymous record (m*, count).

Map: Construct anonymous record m* = p_1, (p_2,….,p_a,sv), p_i, $1 \leq i \leq a$, is the parent of a specialization in current AL and is also an ancestor of v_i in m; emit (m*, count).

Reduce: For each m*, sum □ ∑count; emit (m*, sum)[15].

ANONYMIZATION

Data anonymization can reduce concerns about protection and security and pursue legitimate prerequisites. Anonymization is not insusceptible countermeasures that compromise Current anonymization schemes in discharged databases will reveal promised data. It is sufficient for anonymization after the approval of the individual data sets. We may store or evacuate the touchy area in data sets through anonymization. After that, for the small data sets, it gets the middle of the road result and is used for the halfway results. Data anonymization algorithm that converts clear text data into a nonhuman readable and irreversible form including but not limited to pre-image resistant hashes and encryption techniques in which the decryption key has been discarded.

V. IMPLEMENTATION

We are suggesting a highly versatile two-stage TDS method to cloud-based information anonymization. To use MapReduce's parallel capacity on the internet, two-stage specializations needed in an anonymization phase are necessary. In the first phase, we need to parcel in a collection of littler data sets, and each of these data sets is anonymized in parallel, we get the middle of the results of the road. The middle of the road results are collected into one in the subsequent stage and further anonymized to achieve reliable k-unknown data sets. Here MapReduce is helpful in the two stages of stable computing. A collection of MapReduce occupations is deliberately structured andco-operatively composed to perform data sets specializations. Originally, we imaginatively apply MapReduce to TDS on cloud for information anonymization and purposefully design a series of innovative MapReduce careers to accomplish the specializations in a highly versatile manner. Secondly, we propose a two-stage TDS approach to addressing increased versatility by allowing for parallel execution of specializations on numerous data parcels during the first phase [16].

For example, Table 1 is to be anonymized with Anonymization Level (AL) as set to 2 and QI={ AGE, GENDER, PINCODE, PHONE} as set to Quasi Identifiers. The group distinguishes the semi-identifiers as shown by its standards and guidance.

Table 1: Anonymization level

Name	Age	Gender	Pincode	Phone	Disease
Akash	30	M	590014	9434	Heart Blockage
Boby	40	M	590001	9556	STI
Carman	50	M	592231	9898	Lung Cancer
Denny	60	F	590001	8987	Psoriasis
Eli	75	F	590002	8086	Psoriasis

The NAME characteristic here is "Touchy," so before anonymizing the table above, we might want to "smother" this property. After covering up, Table 2 looks as if it was below

Table 2: AL after suppression

AGE	GENDER	PINCODE	PHONE	DISEASE
30	M	590014	9434	Heart Blockage
40	M	590001	9556	STI
50	M	592231	9898	Lung Cancer

Anonymizing data by means of Top-Down Specialization would add that characteristic value to the base of Taxonomy Tree in Table 3.

Table 3: Root of taxonomy tree

AGE	GENDER	PINCODE	PHONE	DISEASE
[0 - 100]	ANY	******	****	Heart Blockage
[0- 100]	ANY	******	****	STI
[0 - 100]	ANY	******	****	Lung Cancer
[0 - 100]	ANY	******	****	Psoriasis
[0 - 100]	ANY	******	****	Psoriasis

The data in the table above is highly protected, but the utility of the data is extremely low. The information is very private. We note here that data anonymization is not just the one goal we are trying to achieve through anonymization. Therefore, we make sure that the data value is high enough to make the information useful for mining.

The Top-Down Specialization Algorithm must exercise the trait esteems iteratively until the k Anonymization is infringed. The table given in the wake of anonymizing it for k=2 in table 4 would look like

Table 4: Anonymized dataset

AGE	GENDER	PINCODE	PHONE	DISEASE
[0 - 50]	M	5900**	9***	Heart Blockage
[26-50]	M	590001	9***	STI
[26-50]	M	59*****	9***	Lung Cancer
[26-50]	F	590001	8***	Psoriasis
[51-100]	F	59000*	8***	Psoriasis

VI. MAPREDUCE FUTURE RESEARCH DIRECTIONS

Google is familiar with MapReduce's handling of unstructured data, such as cloud records. There are research needs that focus on MapReduce reconsideration to manage organized data and stream data [27]. As stated in this analysis, the work effect of MapReduce can be divided in two different ways, the first approach is used to update the MapReduce programming template. The second course of study is concerned with modifying the current estimates in different areas to be performed in MapReduce. Software engineers who are limited to articulate the equation in outline reduction works should be able to do this so that future research aims to change the flow equations as information for MapReduce. Then another work trend is building up a MapReduce layer that eventually or semi-naturally transforms momentum calculations to be ideal for MapReduce programming design. Scientists can also explore new equations that are concerned about the drawbacks of MapReduce

VII. CONCLUSION

Eventually, we can say we learned from this job and get knowledge about different things. Initially, we worked out how to restrict the programming template, how to make concurrent computations and dispersions, and how to make those computations more tolerant. In fact, coordinating capacity for data transfer is a valuable commodity. Different improvements in our framework are focused on reducing the amount of data sent through the system: region advancement allows us to discover data from nearby plates, spares arrange transfer speed. Second, excess execution can be used to reduce the effect of medium executed devices on stable locations for lead disappointments and data handling.

MapReduce was developed by Google to take care of the large volume of data. We're trying to present the MapReduce programming method in this article. We present capabilities, confines, and present research endeavors here, MapReduce. This is the fundamental reason for improving MapReduce power with its restrictions. We also outlined how MapReduce execution can be upgraded by monitoring data slant. In addition, for groups running MapReduce work, we discussed current power enhancement solutions. We eventually spoke about his future research in this research paper.

Due to the larger size of data sets, confidentiality protection of data analysis, exchange and processing in the cloud environment is difficult.

REFERENCES

1. H. Takabi, J.B.D. Joshi and G. Ahn (2010). “Security and Privacy Callenges in Cloud Computing Environments,”IEEE Security and Privacy, vol. 8, no. 6, pp. 24-31.
2. S. Chaudhuri (2012). “What Next?: A Half-Dozen Data Management Research Goals for Big Data and the Cloud,” Proc. 31st Symp. Principles of Database Systems (PODS ’12), pp. 1-4.
3. Shrivastva K.M.P., Rizvi M.A., Singh S. (2014). Big Data Privacy Based on Differential Privacy a Hope for Big Data. 2014 International Conference on Computational Intelligence and Communication Networks 776–781. DOI: 10.1109/CICN.2014.167
4. N. Mohammed, B.C. Fung, and M. Debbabi (2011). “Anonymity MeetsGame Theory: Secure Data Integration with Malicious Participants,” VLDB J., Vol.20, No. 4, pp. 567-588.
5. L. Kaufman and P. Rousseeuw (1990). Finding Groups in Data An Introduction to Cluster Analysis. New York: Wiley Interscience.
6. Roy, S.T.V. Setty, A. Kilzer, V. Shmatikov, and E. Witchel (2010). “Airavat: Security and Privacy for Mapreduce,” Proc. Seventh USENIX Conf. Networked Systems Design and Implementation (NSDI ’10), pp. 297-312.
7. D. Zissis and D. Lekkas (2011). “Addressing Cloud Computing Security Issues,” Fut. Gener. Comput. Syst., vol. 28, no.3, pp. 583-592.
8. J. Leverich and C. Kozyrakis (2010). "On the Energy (In) efficiency of Hadoop Clusters," ACM SIGOPS Operating Systems Review, vol. 44, no. 1, pp. 61-65.
9. L. Sweeney (2002). “K-Anonymity: A Model for Protecting Privacy,” Int’lJ. Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, no. 5, pp. 557-570.
10. 10.Y. Kwon, M. Balazinska, B. Howe, and J. Rolia (2011). "A study of skew in mapreduce applications," presented in the 5th Open Cirrus Summit.
11. 11.J. Dittrich et. al. (2010). "Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing)," PVLDB, vol. 3, no. 1, pp. 518-529.
12. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Le-Ung (2003). The Google file system. In 19th Symposium on Op- erating Systems Principles, pages 29–43, Lake George, New York.
13. B.C.M. Fung, K. Wang, and P.S. Yu (2007). “Anonymizing Classification Data for Privacy Preservation,” IEEE Trans. Knowledge and Data Eng., vol. 19, no. 5, pp. 711-725.
14. E. Elnikety, T. Elsayed, and H. E. Ramadan (2011). "iHadoop: Asynchronous Iterations for MapReduce," in Proc. the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, pp. 81- 90.
15. Y. Zhang, Q. Gao, L. Gao, and C. Wang (2011). "iMapReduce: A Distributed Computing Framework for Iterative Computation," in Proc. the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum, pp. 1112-1121.
16. Luiz A. Barroso, Jeffrey Dean, and Urs Ho¨lzle (2003). Web search for a planet: The Google cluster architecture. IEEE Micro, 23(2): pp. 22–28.
17. J. Sedayao: Enhancing cloud security using data anonymization, White Paper, Intel Coporation.
18. Top Ten Big Data Security and Privacy Challenges, Technical report, Cloud Security Alliance, November 2012.
19. X. Zhang, L. T. Yang, C. Liu and J. Chen (2014). “A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization using MapReduce on Cloud”, IEEE Transactions on Parallel and Distributed Systems (TPDS), ISSN: 1045-9219. (A*, IF: 1.796), vol. 25, no. 2, pp. 263-373.
20. X. Zhang, C. Liu, S. Nepal, S. Pandey and J. Chen (2012). “A PrivacyLeakage Upper-Bound Constraint Based Approach for Cost-Effective Privacy Preserving of Intermediate Data Sets in Cloud”, IEEE Trans. Parallel and Distributed Systems, to be published.
21. Dean and S. Ghemawat (2010). “Mapreduce: A Flexible Data Processing Tool,” Comm. ACM, vol. 53, no. 1, pp. 72-77.
22. T. Wang, S. Meng, B. Bamba, L. Liu, and C.Pu (2009). A general proximity privacy principle, in Proc. IEEE 25th Int. Conf. Data Eng.
23. G. Aggarwal, R. Panigrahy, T. Feder, D. Thomas, K. Kenthapadi, S. Khuller, and A. Zhu, Achieving anonymity.
24. N. Mohammed, B. Fung, P.C.K. Hung, and C.K. Lee (2010). “Centralized and Distributed Anonymization for High- Dimensional Healthcare Data,” ACM Trans. Knowledge Discovery from Data, Vol. 4, no. 4, Article 18, 2010.
25. 25.M. Stonebraker et. al. (2010). “MapReduce and Parallel DBMSs: Friends or Foes?" Communications of the ACM, vol. 53, no. 1, pp. 64-71.
26. 26.A. Abouzeid, K. Bajda Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin (2009). “Hadoop DB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads," PVLDB, vol. 2, no. 1, pp. 922-933.
27. Gupta, Palak and Tyagi, Nidhi, “Digital security implementation in big data using Hadoop”, International Journal of Research Studies in Computing, Volume 5 Number 1, 3-9, April, 2016.
28. Risha Tabassum, Dr. Nidhi Tyagi, “Hadoop Identity Authentication using Public Private Key Concept”, International Journal of Engineering Trends and Technology, Volume-45, Number-9 -March 2017.

29. Vanisha Mavi, Nidhi Tyagi, "Hadoop's Second Generation –YARN", International Journal of Contemporary Research in Engineering & Technology, Volume7, Issue 1, and ISSN: 2250-0510, 2017.
30. Ramratan Rathore, P.S. Chowdhary, Nidhi Tyagi, "Verification of Data Integrity using Public Auditability and Data Dynamics for Storage Security in Cloud Computing", International Journal of Advance Research In Science And Engineering, 3(5):79-84, 2014.
31. Ashutosh Dixit, Nidhi Tyagi, "Big Data Privacy for End to End Delivery", International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-2S8, August 2019.

AUTHORS PROFILE

Ashutosh Dixit, Research Scholar, from Bhagwant University, Ajmer, Rajasthan, India. He has teaching experience of 8 years, in reputed Engineering College. His area of Interest includes Big Data, DBMS, Data Structure, Cyber Security, & IOT.

Dr. Nidhi Tyagi, working as a Professor in Department of IT at MIET, Meerut, U.P. India.. She has teaching experience of 19 years, in reputed Institutes and Universities. She has published 40 research papers in international and national journals. Her area of interest includes Information Retrieval System, Big Data, and Databases & IOT.