Investigation Analysis On Data Prefetching And Mapreduce Techniques For User Query Processing

S.Tamil Selvan, K.A.Dhamotharan, G.Saravanan, R.Karunamoorthi

Abstract: Big data is a group of data used for examining and extracting the useful information in recent days. Big data analytics is the application of advanced analytic methods against large data comprising structured, semi-structured and unstructured data. The main aim is to store, access and manipulate the data at one place. MapReduce (MR) is the principle of Big Data Processing model with distributed large datasets. Prefetching is a suitable technique for transferring the memory objects in memory hierarchy before required by processor. Many researchers carried out their research on the mapreduce method for responding to the user requested task through removing repeated tasks. But, the job completion time was not reduced using existing techniques as data prefetching was not carried out in existing methods. In order to address these problems, the existing prefetching and map reduce techniques are reviewed and drawbacks of techniques are listed in this paper.

Index Terms: Big data, Job completion, MapReduce, Memory hierarchy, Prefetching, Unstructured data, User Query Processing.

1 INTRODUCTION

Big data is the high-volume, high-velocity and high-variety information that demand cost-effective for decision making. Big data computing has large impact in recent years as MapReduce and cloud computing methods are widespread. MapReduce is the fundamental infrastructure of service when public cloud experienced vulnerability problem. Prefetches are started through explicit fetch operation within the program or through logic that monitors the processor referencing pattern to infer prefetching opportunities. This paper is organized as follows: Section 2 explains the review on different prefetching and mapreduce techniques for user query processing, Section 3 explains the study and analysis of existing prefetching and mapreduce techniques. In section 4, possible comparison of existing methods is made. Section 5 gives the discussion and limitations of the existing prefetching and mapreduce techniques for user query processing are discussed with future direction and Section 6 concludes the paper

2 LITERATURE SURVEY

A prefetching service based task scheduler termed High Performance Scheduling Optimizer (HPSO) was designed in [1] with prefetching to enhance the data locality for MapReduce jobs. The designed method minimized map tasks causing the remote data delay and enhanced Hadoop clusters. But, job completion time was not reduced using HPSO. A Merkle tree-based verification method (MtMR) was introduced in [2] to guarantee high result integrity of MapReduce jobs. MtMR sampled the small portion of task input/output records on the private cloud and performed Merkle tree-based verification on all task input/output records. But, the prefetching was not carried out at earlier stage to minimize the job completion time. A scalable pipeline of components constructed on the Spark engine for large-scale data processing in [3]. The main aim was to collect the data from dataset access logs for organizing them into weekly snapshots and predictive techniques to forecast the datasets. But, the latency was not minimized because prefetching was not carried out. An envisioned future large-scale computing architecture were adapted in [4] for batch processing of big data application in MapReduce model. But, computational complexity was not minimized by future large-scale computing architectures. The big data processing framework was designed in [5] to join the climate and health data and to find the correlation between the climate parameters. But, prefetching was not performed in big data processing framework. A novel intermediate data partition scheme was designed in [6] to reduce the network traffic cost for MapReduce job. Every aggregator reduced the merged traffic from multiple map tasks through addressing the aggregator placement issue. However, the data partitioning was not carried out efficiently through intermediate data partition scheme.High-Level MapReduce Query Languages was built in [7] on MR that converted the queries into an executable native MR jobs. But, the complexity was not minimized through High-Level MapReduce Query Languages. A new A* algorithm introduced in [8] reduced the Map and Reduce tasks for running the path computation on Hadoop MapReduce framework. The designed framework enhanced the feasibility and reliability. A* algorithm minimized computation time. However, MapReduce tasks was not performed by A* algorithm. A novel approach was designed in [9] to improve the metadata management performance for Hadoop in multitenant environment based on the prefetching mechanism. However, the map reduce function was not employed in the multitenant environment. A new efficient pattern mining algorithm was introduced in [10] by using MapReduce framework and Hadoop open-source implementation in the big data. A maximal AprioriMR algorithm was designed for mining condensed frequent patterns. But, the execution time was not minimized using efficient pattern mining algorithm. Secured Map Reduce (SMR) Layer was designed in [11] between the HDFS and MR Layer for improving the security and privacy. The designed model provided the privacy and security through

\

- *S.Tamil Selvan is currently working as Assistant Professor in Computer Science and Engineering in Erode Sengunthar Engineering College, Erode, India,E-mail: stamilselvan@esec.ac.in*
- *K.A.Dhamotharan is currently working as Assistant Professor in Computer Science and Engineering in Erode Sengunthar Engineering College, Erode, India, E-mail: dhamuerode@gmail.com*
- *G.Saravanan is currently working as Assistant Professor in Computer Science and Engineering in Erode Sengunthar Engineering College, Erode, India, E-mail: gsaravanan.esec@gmail.com*
- *R.Karunamoorthi is currently working as Assistant Professor in Computer Science and Engineering in Erode Sengunthar Engineering College, Erode, India, E-mail: karunamoorthir@gmail.com*

resolving scalability problems of privacy for data miners. Though security problem was addressed, prefetching was not carried out in the SMR Layer.

3 PREFETCHING AND MAPREDUCE TECHNIQUES FOR USER QUERY PROCESSING

Prefetching in computer science is a method for increasing the speed of the fetch operations whose result is predictable at the earlier stage. MapReduce is the parallel programming model and employed as distributed system on the cluster. Every request by user is termed as job. Each job is divided into multiple tasks. MapReduce comprised one master and number of workers. The master manages the whole computation through managing the jobs, scheduling the tasks and preserving the load balance. Every job recovered the input data from DFS and stores the result after job completion. Each MapReduce job comprised three phases, namely map phase, shuffle phase and reduce phase. Tasks performed in map and reduce phase are termed as map tasks and reduce tasks respectively. In map phase, input data are collected from DFS and partitioned into many blocks. Each data block allocated to one map task and processed separately. Every map task comprised the collection of records with the <key, value> pairs format. Records in every task result are arranged by the key values. In reduce phase, every reduce task processes map task output records with the particular keys. The map task output allocated depending on the key in record termed shuffle phase. Every reduce task collect their input and perform the reduce function to aggregate input record into reduce task output termed final job output.

3.1 Scheduling Algorithm Based on Prefetching in MapReduce Clusters

High Performance Scheduling Optimizer (HPSO) employed prefetching service based task scheduler to enhance the data locality for MapReduce jobs during prefetching process. The key objective was is to forecast the suitable nodes for future map tasks depending on the current pending tasks and preload the required data to the memory without delay on launching new tasks. The prefetching accuracy was an essential factor that affects the performance. In MapReduce clusters, task scheduler identified the mapping between tasks and nodes. The execution time of map tasks were forecasted and identified the series where the nodes free the busy slots. Based on node sequence, HPSO forecasted and allocated the suitable map tasks to the nodes time. When the scheduling decisions were taken, nodes preloaded associated input data from remote nodes to memory before launching the tasks. An input data prefetching was carried out with data processing where the data transfer overhead was overlapped with the data processing in time dimension. HPSO combined the task scheduler, prediction and prefetching mechanism to develop the data locality and to minimize the network overhead. HPSO was to overlap the data transmission process of next map task with data processing of running map task. The architecture diagram of HPSO was described in Fig 1.

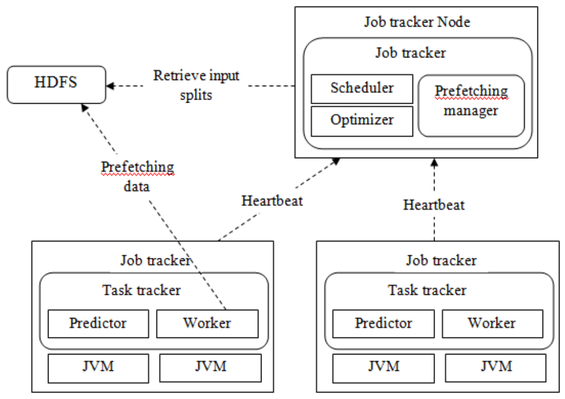

Fig 1. *Architecture Diagram of HPSO*

HPSO comprised three main parts, namely prediction module, scheduling optimizer and prefetching module. Scheduling optimizer forecasted the suitable tasktracker nodes where the future map tasks get allocated. When scheduling decisions were taken before map tasks get scheduled, HPSO activated the prefetching module to load the expected input data. The designed method discovered underutilized disk bandwidth or network bandwidth in CPU-intensive process. The pipelining hided the data transfer latency. The prediction module in every tasktracker node predicted the remaining execution time of map tasks for pipelining and evaluated the sequence where slots become idle. The prefetching module used the master-slave architecture with one prefetching manager in JobTracker and set of workers positioned at the TaskTrackers. Prefetching manager was responsible for monitoring the workers status and for coordinating the prefetching process for map tasks. Each worker completed the loading data block by itself before receiving the map task. When prefetching manager attained prefetching instructions from scheduling optimizer, prefetching manager trigger workers to load data to the memory. Prefetching manager submitted the scheduling optimizer to data blocks in Task Tracker prefetching buffer.

3.2 MtMR: Ensuring MapReduce Computation Integrity with Merkle Tree-based Verifications

Merkle tree-based verification method (MtMR) was introduced to provide better result integrity of MapReduce jobs. MtMR was MapReduce on hybrid cloud environment and used two round of Merkle tree-based verifications on pre-reduce phase as well as reduce phase correspondingly. Depending on MtMR design, theoretical studies were carried out to improve security and overhead performance. MtMR was promising method in terms of the higher integrity and lesser overhead. Merkle tree-based verification framework was introduced to boost the integrity results of the MapReduce computations. MtMR constructed the hybrid cloud architecture based on the benefits of the private cloud and public cloud. The public cloud acquired additional computing and storage resources. The public cloud executed the majority of computation but not guaranteed the result integrity. The private cloud was controlled by computing trusted task owner. The private cloud failed to possess many resources and it used to perform security-critical computations. In MtMR, master and less number of workers termed verifiers was deployed on private cloud. In addition, other workers were deployed on public cloud. Workers on the public cloud completed majority of the work. The master and verifiers on private cloud managed the result integrity. The key objective was to retain the control at home while assigning the resource-intensive computation to the public cloud. Depending on the hybrid cloud architecture, MtMR used Merkle-tree based verifications on various phases of MapReduce job. MtMR used two rounds of Merkle tree-based verification on pre-reduce phase and reduce phase correspondingly. MtMR sampled small portion of task input/output records and performed Merkle tree based verification on all task input/output records. A semi-honest worker not guaranteed successful cheating under MtMR framework. The record error number was functioned with the sampled record ratio. The optimal value for sampled record ratio attained lowest error number under particular security restrictions.

3.3 Dataset Popularity Predictions for Caching of CMS Big Data

Predictive models based on statistical learning techniques are suitable for studying the performance and scalability of complex computing infrastructures. The training process requires abstract features from variety of measurements collected through historical logging activities and to devise relevant metrics for estimating the behavior of the system under investigation. Log analysis related to the processing of structured or unstructured information collected through several layers of monitoring architectures is a promising research field. The scalable dataset popularity prediction pipeline was employed to highlight the process chain from the raw data ingestion and preparation step up to the ML component producing the machine learned model. The designed model was introduced by PPC strategy driving dataset caching in various CMS sites. The main components were described for implementing the pipeline for data preparation and popularity classifier training. A scalable data mining pipeline on CMS Hadoop data store was employed to forecast the popularity of new datasets accessed by jobs processing any event types stored in distributed CMS infrastructure. The dataset accesses problem were casted to binary classification problem. The predictive models improved the accuracy denoting the ability to separate datasets from unpopular ones. A new intelligent data caching policy termed Popularity Prediction Caching (PPC) was carried out to attain the popularity predictions by classifier for optimizing the eviction policy at every site of CMS infrastructure. The efficiency of caching policy was computed through measuring the hit rates attained by PPC and caching the baselines like Least Recently Used (LRU) in handling dataset access requests.

4 PERFORMANCE ANALYSIS OF PREFETCHING AND MAPREDUCE TECHNIQUES FOR USER QUERY PROCESSING

In order to compare the prefetching and mapreduce techniques for user query processing, number of user requested tasks is taken to conduct the experiment. Various parameters are used for improving the user query processing using prefetching and mapreduce techniques.

4.1 Job Completion Time

Job completion time is defined as the amount of time taken for completing the jobs. It is the difference of starting time and ending time of job completion. It is measured in terms of milliseconds. It is formulated as,

$$\textit{Job Completion Time} = \textit{Ending time} - \textit{Starting time of job completion} \qquad (1)$$

From (1), job completion time is calculated. When the job completion time is lesser, the method is said to be more efficient.

TABLE 1
TABULATION FOR JOB COMPLETION TIME

Number of user requested tasks (Number)	*Job Completion Time (ms)*		
	HPSO	*MtMR Method*	*Scalable dataset popularity prediction pipeline*
10	*17*	*28*	*35*
20	*20*	*29*	*37*
30	*22*	*32*	*39*
40	*19*	*30*	*36*
50	*17*	*27*	*34*
60	*18*	*29*	*37*
70	*21*	*31*	*40*
80	*24*	*33*	*42*
90	*27*	*36*	*45*
100	*30*	*39*	*48*

Table 1 explains the job completion time performance with respect to the number of user requested tasks ranging from 10 to 100. Job completion time comparison takes place on existing High Performance Scheduling Optimizer (HPSO), Merkle tree-based verification method (MtMR) and scalable dataset popularity prediction pipeline. The graphical representation of job completion time is explained in Fig 2.

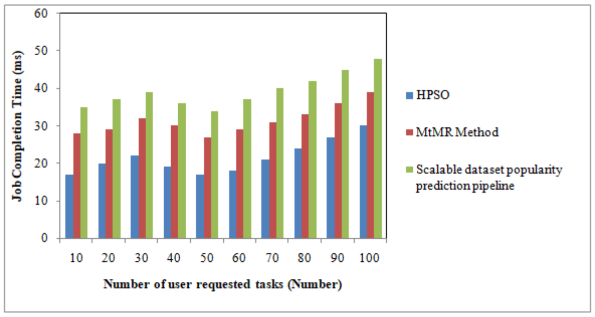

Fig 2. *Measure of Job Completion Time*

From Fig 2, job completion time for different number of user requested task is described. It is clear that the job completion time using High Performance Scheduling Optimizer (HPSO) is lesser when compared to the scalable dataset popularity prediction pipeline and Merkle tree-based verification method (MtMR). This is because HPSO forecasted and allocated the appropriate map tasks to the node time. When scheduling decisions were taken, nodes preloaded the associated input data from remote nodes or local disk to memory before launching tasks. HPSO combined the task scheduler, prediction and prefetching mechanism to develop data locality and to reduce the network overhead. HPSO overlapped the data transmission process of next map task with data processing of running map task. The job completion time of High Performance Scheduling Optimizer (HPSO) is 46% lesser than scalable dataset popularity prediction pipeline and

32% lesser than Merkle tree-based verification method (MtMR).

4.2 Error Rate

Error rate is defined as the ratio of number of user requested tasks that are incorrectly predicted to the total number of user tasks. It is measured in terms of percentage (%). It is formulated as,

$$Error\ Rate = \frac{Number\ of\ user\ tasks\ that\ are\ incorrectly\ predicted}{Total\ number\ of\ user\ tasks} \quad (2)$$

From (2), the error rate is determined. When the error rate is lesser, the method is said to be more efficient.

TABLE 2
TABULATION FOR ERROR RATE

Number of user requested tasks (Number)	Error Rate (%)		
	HPSO	MtMR Method	Scalable dataset popularity prediction pipeline
10	25	18	39
20	27	21	42
30	24	20	38
40	21	17	35
50	19	15	32
60	20	16	34
70	18	14	33
80	21	17	35
90	23	19	37
100	24	21	39

Table 2 describes the error rate performance with respect to the number of user requested tasks ranging from 10 to 100. Job completion time comparison takes place on existing High Performance Scheduling Optimizer (HPSO), Merkle tree-based verification method (MtMR) and Scalable dataset popularity prediction pipeline. The graphical representation of error rate is described in Fig 3.

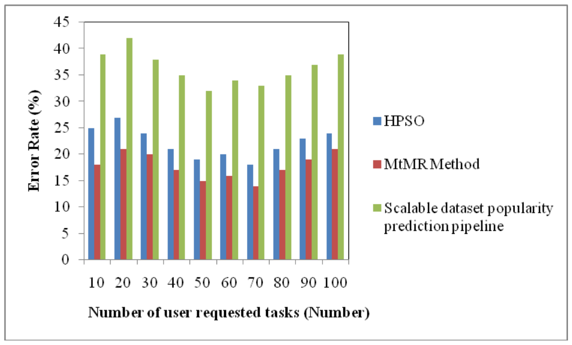

***Fig 3.** Measure of Error Rate*

From Fig 3, error rate for different number of user requested task is described. It is observed that the error rate using Merkle tree-based verification method (MtMR) is lesser when compared to the High Performance Scheduling Optimizer (HPSO) and scalable dataset popularity prediction pipeline. This is because of using Merkle tree-based verification framework to improve the integrity performance of MapReduce computations. MtMR built hybrid cloud architecture depending on the advantages of private cloud and public cloud. MtMR sampled small portion of reduce task input/output records and performed the Merkle tree based verification on all task input/output records. The error rate of Merkle tree-based verification method (MtMR) is 20% lesser than High Performance Scheduling Optimizer (HPSO) and 51% lesser than scalable dataset popularity prediction pipeline.

4.3 Classification Accuracy

Classification accuracy is defined as the ratio of number of user tasks that are correctly classified to the total number of user requested tasks. It is measured in terms of percentage (%). It is given by,

$$Classification\ accuracy = \frac{Number\ of\ user\ tasks\ that\ are\ correctly\ classified}{Total\ number\ of\ usder\ tasks} \quad (3)$$

From (3), the classification accuracy is calculated. When the classification accuracy is higher, the method is said to be more efficient.

TABLE 3
TABULATION FOR CLASSIFICATION ACCURACY

Number of user requested tasks (Number)	Classification Accuracy (%)		
	HPSO	MtMR Method	Scalable dataset popularity prediction pipeline
10	75	88	95
20	72	86	92
30	70	84	90
40	68	81	88
50	65	79	86
60	62	77	84
70	60	75	82
80	58	72	79
90	55	69	76
100	52	66	72

Table 3 explains the classification accuracy performance with respect to the number of user requested tasks ranging from 10 to 100. Classification accuracy comparison takes place on existing High Performance Scheduling Optimizer (HPSO), Merkle tree-based verification method (MtMR) and Scalable dataset popularity prediction pipeline. The graphical representation of classification accuracy is illustrated in Fig 4.

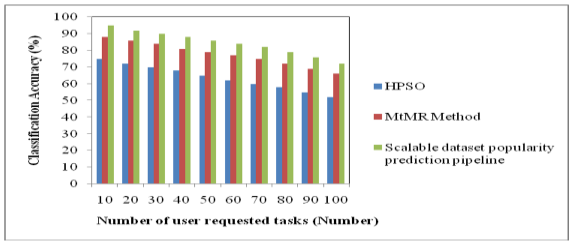

***Fig 4**. Measure of Accuracy*

From Fig 4, classification accuracy for different number of user requested task is explained. It is observed that the classification accuracy using scalable dataset popularity prediction pipeline is higher when compared to the High Performance Scheduling Optimizer (HPSO) and Merkle tree-based verification method (MtMR). This is because of using scalable data mining pipeline on CMS Hadoop data store to forecast popularity of new datasets accessed by jobs processing for any event type stored in distributed CMS infrastructure. Popularity Prediction Caching (PPC) attained the popularity predictions through classifier for optimizing the eviction policy at each CMS infrastructure. The classification accuracy of scalable dataset popularity prediction pipeline is 33% higher than High Performance Scheduling Optimizer (HPSO) and 9% higher than Merkle tree-based verification method (MtMR).

5 DISCUSSION AND LIMITATION ON PREFETCHING AND MAPREDUCE TECHNIQUES FOR USER QUERY PROCESSING

HPSO predicted the suitable nodes for future map tasks depending on pending tasks and preload the required data to memory without any delay on launching new tasks. HPSO minimized the map tasks resulting in remote data delay and enhanced the Hadoop clusters performance. HPSO developed the task scheduler to preload essential input data for launching tasks to the TaskTracker. It minimized the waiting period of map tasks with rack and rackoff locality. A scheduling optimizer was incorporated into the HPSO to enhance the prefetching rate. But, the job completion time was not minimized using HPSO. Merkle tree-based verification method guaranteed high integrity results of MapReduce jobs. Semi-honest worker not performed safe cheating by MtMR framework. MtMR improved the integrity results while acquiring moderate performance overhead. MtMR employed the hybrid cloud architecture for Merkle-tree based verification to guarantee high integrity on the job results. However, prefetching was not carried out at earlier stage to reduce the job completion time. Scalable pipeline of components was constructed on the Spark engine for large-scale data processing. It collected the data from different areas into weekly snapshots for forecasting purpose. The high accuracy represented ability of learned model to separate the popular datasets from unpopular ones. CMS data placement policy has significant improvement of resource usage and resultant reduction of large cost. But, the latency was not reduced because prefetching was not performed.

5.1 Future Direction

The future direction of the work is to perform the user query processing through prefetching and mapreduce function by using machine learning and ensemble learning techniques with higher accuracy and lesser time consumption.

6 CONCLUSION

The comparison of different existing prefetching and mapreduce techniques for user query processing is carried out in this paper. From the survival study, it is clear that the latency was not reduced because prefetching was not performed. The review explains that prefetching was not carried out at earlier stage for minimizing the job completion time. In addition, the classification accuracy was not improved. The wide range of experiments on existing techniques describes the performance of many prefetching and mapreduce techniques with its limitations. Finally from the result, research work can be carried out using machine learning techniques for minimizing job completion time and for improving the accuracy during user query processing Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions. Authors are strongly encouraged not to call out multiple figures or tables in the conclusion—these should be referenced in the body of the paper.

7 REFERENCES

[1] Mingming Sun, Hang Zhuang, Xuehai Zhoua, Kun Lu and Changlong Li, "Scheduling Algorithm based on Prefetching in MapReduce Clusters", Applied Soft Computing, Elsevier, Volume 38, January 2016, Pages 1109-1118

[2] Yongzhi Wang, Yulong Shen, Hua Wang, Jinli Cao and Xiaohong Jiang, "MtMR: Ensuring MapReduce Computation Integrity with Merkle Tree-based Verifications", IEEE Transactions on Big Data, Volume 4, Issue 3, September 2018, Pages 418 – 431

[3] Marco Meoni, Raffaele Perego and Nicola Tonellotto, "Dataset Popularity Prediction for Caching of CMS Big Data", Journal of Grid Computing, Springer, Volume 16, Issue 2, June 2018, Pages pp 211–228

[4] M. Goudarzi, "Heterogeneous Architectures for Big Data Batch Processing in MapReduce Paradigm", IEEE Transactions on Big Data, Volume 5, Issue 1, March 2019, Pages 18 – 33

[5] Gunasekaran Manogaran, Daphne Lopez and Naveen Chilamkurti, "In-Mapper Combiner based Map-Reduce Algorithm for Big Data Processing of IoT based Climate Data", Future Generation Computer Systems, Elsevier, Volume 86, September 2018, Pages 433-445

[6] Huan Ke, Peng Li, Song Guo, and Minyi Guo, "On Traffic-Aware Partition and Aggregation in MapReduce for Big Data Applications", IEEE Transactions on Parallel and Distributed Systems, Volume 27, Issue 3, March 2016, Pages 818 - 828

[7] Marouane Birjali, Abderrahim Beni Hssane and Mohammed Erritali, "Evaluation of high‐level query languages based on MapReduce in Big Data", Journal of Big Data, Springer, Volume 5, Issue 36, December 2018, Pages 2-21

[8] Wilfried Yves Hamilton Adoni, Tarik Nahhal, Brahim Aghezzaf and Abdeltif Elbyed "The MapReduce‐based approach to improve the shortest path computation in large‐scale road networks: the case of A* algorithm", Journal of Big Data, Springer, Volume 5, Issue 16, December 2018, Pages 1-24

[9] Minh Chau Nguyen, Heesun Won, Siwoon Son, Myeong-Seon Gil and Yang-Sae Moon, "Prefetching-based metadata management in Advanced Multitenant Hadoop" The Journal of Supercomputing, Springer, Volume 75, Issue 2, February 2019, Pages 533–553

[10] Jose Maria Luna, Francisco Padillo, Mykola Pechenizkiy and Sebastian Ventura, "Apriori Versions Based on MapReduce for Mining Frequent Patterns on Big Data", IEEE Transactions on Cybernetics, Volume 48, Issue 10, October 2018, Pages 2851 – 2865

[11] Priyank Jain, Manasi Gyanchandani and Nilay Khare, "Enhanced Secured Map Reduce layer for Big Data privacy and security", Journal of Big Data, Springer, Volume 6, Issue 30, 2019, Pages 1-17