ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)

Vol. 2, Special Issue 10, March 2016

INCREMENTAL STREAMING DATA FOR MAPREDUCE IN BIG DATA USING HADOOP

S.Kavina[1], P.Kanmani[2]
P.G.Scholar, CSE, K.S.Rangasamy College of Technology, Namakkal, Tamil Nadu, India[1]
askkavina@gmail.com[1]
Assistant Professor, CSE, K.S.Rangasamy College of Technology, Namakkal, Tamil Nadu, India[2]
pkanmaniit@gmail.com[2]

Abstract

Big data is used to handle the large amount of data using hadoop. Today, the Data mining application result become stale and obsolete over time due to an arrival of the new data and updates in many important areas such as e-commerce, health care, social network, education and environment. In Exiting, incremental processing is very promising approach to refreshing the mining result. The proposed work of i^{2}MapReduce is an extension to Map Reduce that supports fine-grain incremental processing for both one step incremental processing and iterative computation. In this computation, three algorithms are mainly used: i) A novel Fine-grain incremental processing using MRBG-Store, ii) Collaborative clustering approach algorithm and iii) a novel incremental processing. This algorithm increases the efficiency of the incremental processing. In order to provide good performance and fast analysis in any data set such as e-commerce, health care, social network and education are handled by using Hadoop.

Keywords: ***Big data, incremental processing, iterative calculation, mapreduce, hadoop.***

I. Introduction

Streaming data to be incremented once fix the data set in Hadoop Distributed File System (HDFS). During that time, incremental map reduce big data mining evaluating the incremental processing. The Incremental Streaming data for Map Reduce in big data using hadoop through Hadoop environment incremental data to be automatically added mapper and reducer to working incremental data. Thus the response time of the system is very fewer and it works very fast.

1.1 Purpose

A novel incremental iterative processing is significantly more difficult than incremental in a step process, because even a small number of updates to influence can circulate a large part of the Midway states after a number of iterations. The MRBG-store supports the preservation and retrieval of fine-grained MRBGraph states for incremental processing. A novel incremental processing is a hopeful approach to inspirational mining results.

Map Reduce is a programming model and an associated implementation for processing and generating large data sets. Users denote a map function that processes a key/value pair to create a set of in-between key/value pairs, and a reduce function that merges all halfway values correlated with the same intermediate key.

A Map Reduce system (e.g. Apache Hadoop MapReduce) receives the input data of the calculation and writes the final results for a Distributed File System (e.g. Between HDFS), the shares a file in the same size (e.g. , 64 MB) blocks and saves the building blocks over a cluster of machines. For a program Map Reduce, the Map Reduce system runs a job Tracker process on a master node to monitor the progress and a number of task Tracker processes on workers to perform the actual node map and reduce tasks.

The Job Tracker starts a Map Task per data block, and in the rule maps the Task Tracker on the machine stops the corresponding data block to minimize the communication overhead. I^{2}MapReduce expected delta input data, the just inserted, deleted, changed, or kv-pairs as input for incremental processing. The goal of many incremental data acquisition or incremental crawling techniques has been developed to better data collection performance.

1.2 Scope of project

To propose i^{2}MapReduce an extension to MapReduce that support novel fine-grain incremental processing for both one step incremental processing and iterative computation. The proposed work of i^{2}MapReduce includes the following three features: i) Novel Fine-grain incremental processing using MRBG-Store, ii) Collaborative clustering approach algorithm and iii) a novel incremental processing. It supports incremental data input, incremental data processing, intermediate state defence, incremental map and reduce functions. The input manager can dynamically find out new inputs and then give in jobs to the master node.

1.3 Overview

The main objectives of the incremental streaming data for map reduce in big data using hadoop is to improvement and updating of the existing system by increasing the efficiency and use. The software improves the working methods by replacing the existing system with Hadoop incremental data-based system. Big data is being constantly developed. As new data are collected the input data and updates a large data mining algorithm will gradually change and the calculated results are outdated and obsolete in the course of time. Incremental processing is a hopeful approach to refreshing mining results [21]. In view of the size of the input large amounts of data, it is often very expensive, again the entire calculation of reason.

II. Related Works

Yanfeng Zhang et al., describes that iMapReduce significantly improves the performance of iterative algorithms (1) Reduction of overheads to create a new task in each iteration, (2) abolition of redistribution of static data in the shuffle stage of MapReduce, and (3), causing the asynchronous execution of each iteration [20]. MapReduce has become extremely popular is for the analysis of large quantities of data. It offers a simple programming framework and is responsible for the distributed execution of calculations, fault tolerance and load balancing. In MapReduce, the captain shall notify a job in many small tasks, the number of tasks can be executed at the same time cannot be greater than the available task slots. It provides support for the elimination of the redistribution of static data between tasks.

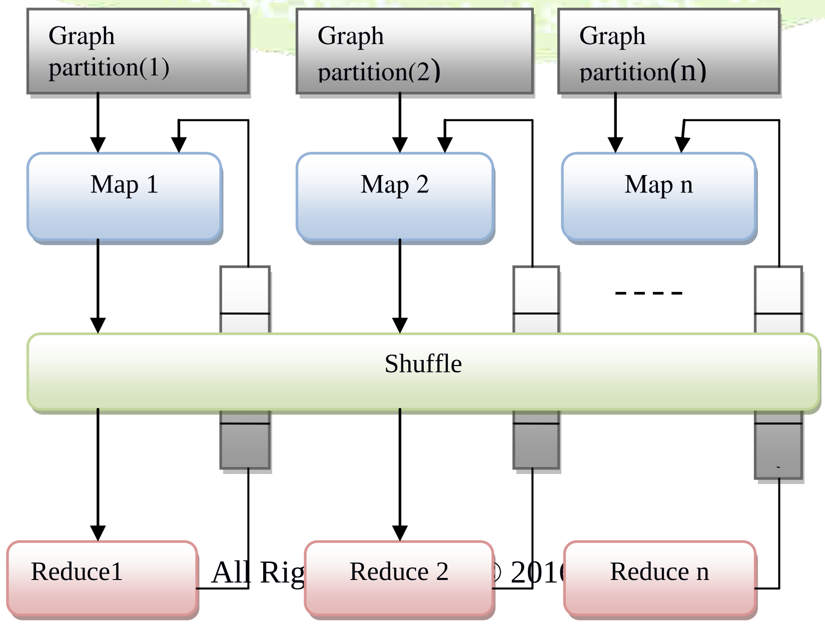

Figure 1: Iterative Processing Structure

Pramod Bhatotia et al., explains online data sets grow gradually in the course of time as new entries are slowly added and existing entries are deleted or changed. Here present a system called Incoop, which enables existing MapReduce programs not for incremental processing, to transparently in an incremental manner [15]. In Incoop, calculations can respond automatically and efficiently on changes to your data input by the re-use of intermediate results from earlier calculations and step-by-step Update of the output according to the changes in the input. The aim is to create a system for large-scale data processing, is able to recognize the performance benefits of incremental calculations, while the complexity of the application and development effort is low.

Yanfeng Zhang et al, showed the calculations are frequently used in the variety of data mining algorithms. PageRank can be used for computing PageRank scores [10]. Iterative algorithms usually stop if an abort condition is fulfilled. To stop a iterative calculation, PrIter offers three alternative methods to do check termination. 1) Distance-based termination check. After each iteration, each worker sends the sum of the values for the master (State the sum operation is performed together with the priority queue extraction). The Master collects these values of various employees and it will stop the iteration if the difference between the added values of two consecutive iterations is less than a threshold. 2) Check sub pass-based termination. Parsing massive data-set iteratively is a time-consuming process.The prioritized execution of iterative calculations accelerates iterative algorithms [10].

III. Incremental MapReduce

Handling Incremental MapReduce

To describe the incremental MapReduce part of the infrastructure by separately considering how Map and Reduce tasks handle incremental inputs.

Scheme1: Map Task

Given that Inc-HDFS already provides the basic rule over the arrangement and granularity of the input parts that are provided to Map tasks, the job of the incremental Map tasks becomes basic, from the time when they can execute task-level memorization without having to worry about finding finer-grained misaligned, or omitted opportunities for reusing previous results. Explicitly, after a Map task runs, store its results constantly and insert a corresponding reference to the result at the memorization server.

Scheme2: Reduce Task

The Reduce function processes the output of the Map function grouped by the keys of the generated key-value pairs. More specifically, for a subset of all keys, each Reduce retrieves the key-value pairs generated by all Map tasks and applies the Reduce function. To ensure efficient memorization of Reduce tasks, to perform memorization at two levels: first as a novel-grained memorization of entire Reduce tasks, and second as a fine-grained memorization of novel abbreviation tasks as described.

As with Map tasks, remember the results of a Reduce task by storing patiently and locally their result and by inserting a mapping from a collision-resistant hash of the input to the location of the output in the memorization server. Since a Reduce task receives input from n Map tasks, the key stored in the memorization server consists of the hashes of the outputs from all n Map tasks that collectively form the input to the Reduce task.

IV. Proposed Work

The incremental streaming data for mapreduce in big data using hadoop is used to improving the performance and efficiency. There are two process in incremental streaming data for mapreduce namely dynamic incremental processing and iterative processing for MapReduce.

4.1 Dynamic Incremental Processing

Compared to previous solutions, i2MapReduce includes the following four characteristics:

(i) Novel fine-grained incremental processing with MRBG-store

In contrast to Incoop, i2MapReduce supports the KV-pair level novel fine-grained incremental processing to the level of the calculation back as much as possible. Our model of the KV-pair level data flow and data dependency in a MapReduce also as a two-sided graphics, called MRBGraph. A MRBG-store is for the conservation of the novel fine-grained states in the MRBGraph and sustains competent queries to retrieve fine-grained states for novel incremental processing.

(ii) Novel incremental processing for the iterative calculation.

A novel incremental iterative processing is radically more difficult than incremental in a step process, because even a small number of updates can propagate to affect a large part of the intermediate steps after a certain number of iterations. To resolve this issue, suggest that the re-use of converged condition from the previous calculation and employ a Change propagation Control (CPC)–mechanism. Also improve the MRBG-store to better support the access patterns in incremental iterative processing was shown in figure 2.

(iii) Clustering-based Collaborative Filtering

Clustering-based collaborative filtering (ClubCF), also called as Clustering and collaborative filtering. Clustering is a pre-processing step to divide, large quantities of data into manageable parts. A cluster contains some similar services like a club contains some like-minded users. This is a further reason alongside the abbreviation that this ClubCF. Since the number of services in a group is much lesser than the total number of services on the network. The computing time of the algorithm can radically reduce CF.

Stage 1: Clustering Stage

1. The words will be stemmed in Dt and Dj using Porter Stemmer. The stemmed words in Dt are set into Dt' and the stemmed words in Dj are put into Dj' .
2. Compute $D\_(st,$ sj), and $F\_sim(st, sj)$ using
   Jaccard similarity coefficient is individually.
3. Calculate $C\_(st,$ sj) by weighted sum ofD_sim(st,sj) and F_sim(st,sj). Construct a matrix D each record of which is a characteristic similarity.
4. Cluster the services rendering to their characteristic similarities in D using an agglomerative hierarchical clustering algorithm.

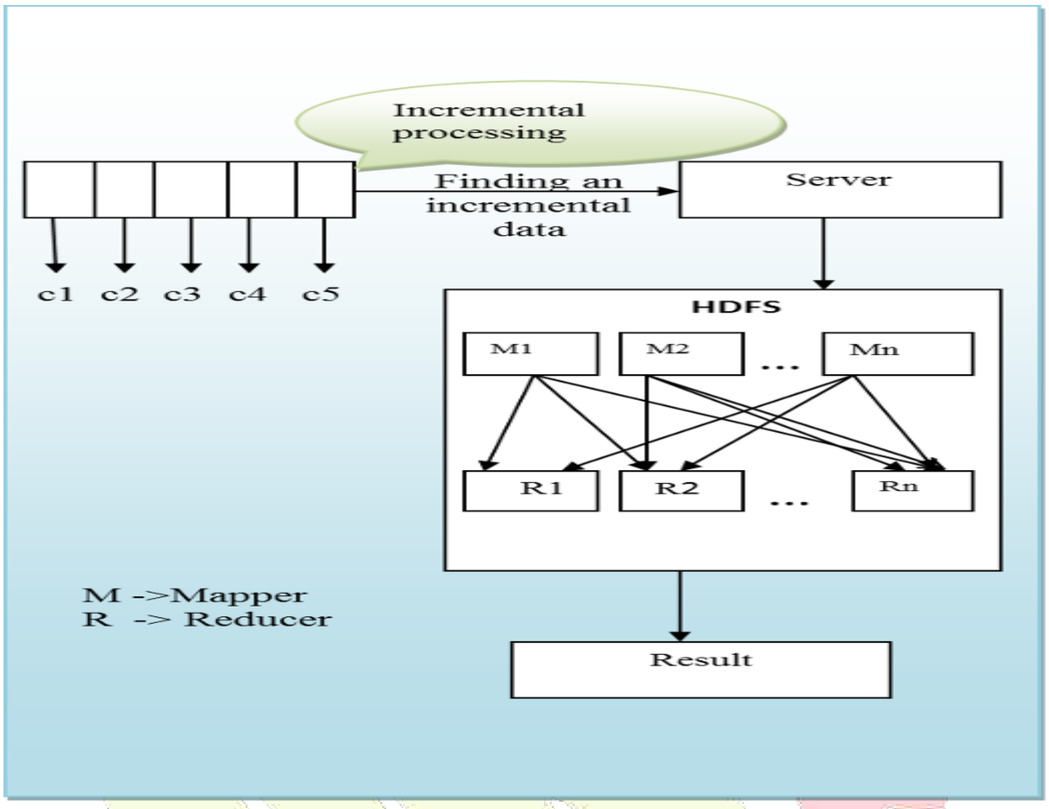

Figure 2: Architecture of Incremental Streaming Data for MapReduce in HDFS

Stage 2: Collaborative filtering Stage

1. Calculate the *R_sim*(*st*,sj) using Pearson correlation coefficient if *st* and *sj* belong to the similar cluster, and calculate *R_sim'*(*st*,sj) by weighting *R_sim*(*st*,*sj*).
2. Select services whose boosted rating similarity with*st* exceed a rating similarity threshold γ, and put them into a neighbours set.
3. Compute the expected rating of *st* for an active user. If the expected rating exceeds a recommending threshold, it will be suggested to the active user.

The idea of a ClubCF method for big data applications related to service recommendation is explained. Services are merged into some clusters via an AHC algorithm before applying any CF technique. After that the rating likenesses between services within the same cluster will be calculated. Moreover, as the ratings of services in the same cluster are more related with each other than with the ones in other clusters. The prediction based on the ratings of the services in the same cluster will be more correct than that of based on the ratings of all similar or dissimilar services in all clusters.

4.2 Iterative Processing for MapReduce

Many algorithms for the analysis of the data and the machine learning use an iterative process. In this section, start with two examples of iterative algorithms, and then together, the limitations for the implementation of these algorithms in MapReduce. I^{2}MapReduce support iterative processing by directly passing the Reduce outputs to Map and by distinguishing variant state data from the static data. Iterative algorithm examples present the Single Source Shortest Path (SSSP) and PageRank algorithms, together with their MapReduce implementations in this section [Figure1].

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

1) Single Source Shortest Path: Single Source Shortest Path (SSSP) is a classical problem that derives the shortest distance from a source node s to all other nodes in a graph [19]. Formally, given a weighted, directed or undirected graph G = (V,E.W), where V is the set of nodes, E is the set of edges, and W(i, j) is a positive weight of the edge from node i to node j. The shortest distance from the source node s to a node j can be computed iteratively as follows:

$$D^{(k)}(j) = \min\{D^{(k-1)}(j), \min_i\{D^{(k-1)}(i) + W(i, j)\}\}$$

Where k is the number of iteration, and i is an incoming neighbor of node j. Initially, $D^0(s) = 0$, and $D^0(j) = 1$ for any node j other than s. Formally, the prioritized SSSP can be described by the MapReduce programming model as follows:

Map: Compute $D(i) + W(i,j)$ for node *i*, send the result to its neighboring node *j*.

Reduce: Select the minimum value among node *j*'s current $D(j)$ and all the results received by *j*, and update $D(j)$ with the minimum value.

Priority: Node j is eligible for the next map operation only if $D(j)$ has changed since the last map operation on *j*.Priority is given to the node *j* with smaller value of $D(j)$ [19].

2) PageRank: PageRank is an algorithm for calculating the importance of nodes in a graph. It was widely used in applications such as Web search, link prediction. Similar algorithms, such as PageRank and rooted the average algorithm have found applications in personalized news and video recommendation systems.

Algorithm 1: PageRank in MapReduce

```
        Map Phase input: $< i, N_i | R_i >$
1.      output $<i, N_i>$
2.      for all j in $N_i$ do
3.          $R_{i,j} = R_i / N_i$
4.          output $<j, R_{i,j}>$
5.      end for
        Reduce Phase input: $< j, \{R_{i,j}, N_j\} >$
6.      $R_j = d\sum i\, R_{i,j} + (1 - d)$
7.      Output $<j, N_j | R_j>$
```

Proposed Design

The following scheme are describing about the incremental streaming data for mapreduce.

Client Model

Client request the big data in to HDFS server to get the data and read it. Receiving dynamic outsources from the external memories and distributing outsources to Mapper and Reducer.

Server Model

Server model gets the requested data set and read the data to load worker node to how many node should work mapping and how many nodes should work reduce job to allocate.

Worker node

This module has takes over the work from server module which is allocated work and processed the job. Worker node should display the result to MapReduce.

MapReduce

In MapReduce, mapper should collect mapped result and reducer should reduce the efficient result to describe shortly.

Report

Finally, report module can received the result from server. Significantly reduce the run time for refreshing big data mining results to re-computation on i^{2}mapreduce.

V. Experimental Analysis

Iterative Processing

The following Table 1 describes experimental result for proposed system algorithms. The table input text size for HDFS and Incremental Processing HDFS algorithms execution time details are shown in the Incremental Processing HDFS Performance Table.

Table 1: Performance of HDFS and Incremental Processing HDFS

Version	Skip offset [MB]	Throughput [MB/S]
HDFS	0.8 0.9 0.97 0.99 1.15	0.4
Incremental Processing HDFS	1.0 1.1 1.0 1.05 1.2	0.8

The following Figure 3 describes experimental result for proposed system algorithms. The table input text size for **HDFS scheduler and incremental processing scheduler algorithms** for encryption execution time details are shown.

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

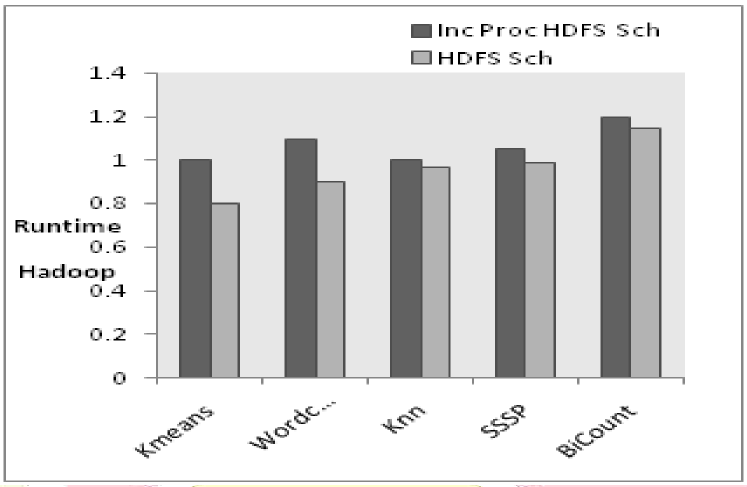

Figure 3: Execution time of Incremental Processing

Incremental Processing

Performance analysis between fine grain incremental processing and a novel fine Grain incremental processing speed up chart in Hadoop for encryption execution time details are shown in the figure 4. A novel fine grain incremental processing algorithm has compared to fine grain incremental processing algorithm, through this chart analyzed the performance. By this chart, throughput has been evaluated capably.

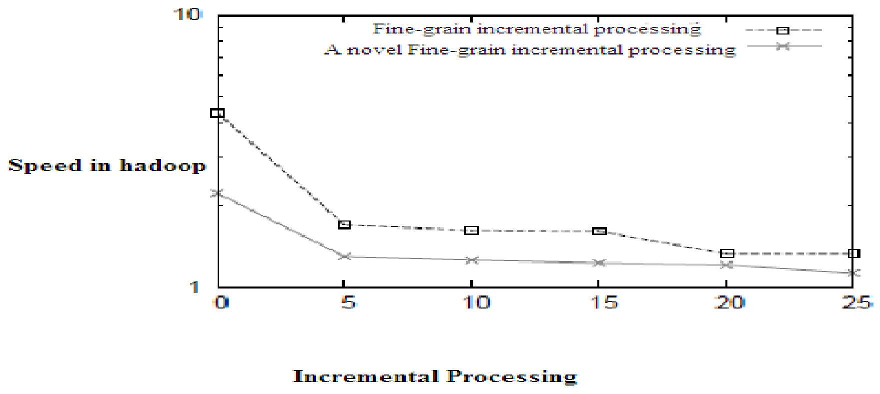

Figure 4: Performance analysis of novel fine grain incremental processing

VI. Conclusion

To describe i^{2}MapReduce, a MapReduce based framework for incremental big data computing. I2MapReduce combines a fine-grained, incremental engine, a general iterative model and a number of effective techniques for incremental iterative calculation. Real-Machine experiments show that i2MapReduce can significantly reduce the duration for a refreshing large data mining results in comparison to re-calculate on both simple and iterative MapReduce. This algorithm increases the efficiency of the incremental processing. MapReduce is the most widely used in production because of its simplicity, generality, and maturity. These papers have to improve the MapReduce performance and also data collections.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in Proc. 6th Conf. Symp. Opear. Syst. Des. Implementation, 2004, p. 10.

[2] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for, in-memory cluster computing," in Proc. 9th USENIX Conf. Net. Syst. Des. Implementation, 2012, p. 2.

[3] R. Power and J. Li, "Piccolo: Building fast, distributed programs with partitioned tables," in Proc. 9th USENIX Conf. Oper. Syst. Des. Implementation, 2010, pp. 1–14.

[4] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 135–146.

[5] S. R. Mihaylov, Z. G. Ives, and S. Guha, "Rex: Recursive, deltabased data-centric computation," in Proc. VLDB Endowment, 2012, vol. 5, no. 11, pp. 1280–1291.

[6] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: A framework for machine learning and data mining in the cloud," in Proc. VLDB Endowment, 2012, vol. 5, no. 8, pp. 716–727.

[7] S. Ewen, K. Tzoumas, M. Kaufmann, and V. Markl, "Spinning fast iterative data flows," in Proc. VLDB Endowment, 2012, vol. 5, no. 11, pp. 1268–1279.

[8] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "Haloop: Efficient iterative data processing on large clusters," in Proc. VLDB Endowment, 2010, vol. 3, no. 1–2, pp. 285–296.

[9] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, "Twister: A runtime for iterative mapreduce," in Proc. 19th ACM Symp. High Performance Distributed Comput., 2010, pp. 810–818.

[10] Y. Zhang, Q. Gao, L. Gao, and C. Wang, "imapreduce: A distributed computing framework for iterative computation," J. Grid Comput., vol. 10, no. 1, pp. 47–68, 2012.

[11] S. Brin, and L. Page, "The anatomy of a large-scale hypertextual web search engine," Comput. Netw. ISDN Syst., vol. 30, no. 1–7, pp. 107–117, Apr. 1998.

[12] D. Peng and F. Dabek, "Large-scale incremental processing using distributed transactions and notifications," in Proc. 9th USENIX Conf. Oper. Syst. Des. Implementation, 2010, pp. 1–15.

[13] D. Logothetis, C. Olston, B. Reed, K. C. Webb, and K. Yocum, "Stateful bulk processing for incremental analytics," in Proc. 1st ACM Symp. Cloud Comput., 2010, pp. 51–62.

[14] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi, “Naiad: A timely dataflow system,” in Proc. 24th ACM Symp. Oper. Syst. Principles, 2013, pp. 439–455.

[15] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin, “Incoop: Mapreduce for incremental computations,” in Proc. 2nd ACM Symp. Cloud Comput., 2011, pp. 7:1–7:14.

[16] J. Cho and H. Garcia-Molina, “The evolution of the web and implications for an incremental crawler,” in Proc. 26th Int. Conf. Very Large Data Bases, 2000, pp. 200–209.

[17] Y. Zhang, S. Chen, Q. Wang, and G. Yu, “i2mapreduce: Incremental mapreduce for mining evolving big data,” CoRR, vol. abs/ 1501.04854, 2015.

[18] Y. Zhang, Q. Gao, L. Gao, and C. Wang, “Priter: A distributed framework for prioritized iterative computations,” in Proc. 2nd ACM Symp. Cloud Comput., 2011, pp. 13:1–13:14.

[19] Y. Bu, V. Borkar, J. Jia, M. J. Carey, and T. Condie, “Pregelix: Big (ger) graph analytics on a dataflow engine,” in Proc. VLDB Endowmen, 2015, vol. 8, no. 2, pp. 161–172.

[20] C. Yan, X. Yang, Z. Yu, M. Li, and X. Li, “IncMR: Incremental data processing based on mapreduce,” in Proc. IEEE 5th Int. Conf. Cloud Comput., 2012, pp. 534–541.

[21] Yanfeng Zhang, Shimin Chen, Qiang Wang, and Ge Yu, Member, IEEE,”i^{2}MapReduce: Incremental MapReduce for Mining Evolving Big Data”, IEEE Transactions on Knowledge and Data Engineering, vol.27, no.7, July-2015.

[22] Kavina.S**,** A.Gnanabaskaran,” A Survey Paper: Incremental Streaming Data For MapReduce In Big Data Using Hadoop”, International Journal Of Applied Engineering Research, Vol.10, no. 85, December-2015, pp.176-181.