INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 7 Issue: 3(2016)

An Efficient Approach For Processing Bigdata with Incremental and Iterative Mapreduce

Shaik Moinuddin Ahmed,
Student, M.Tech, Department ofComputer Science and Systems Engineering, Andhra University, Vishakapatnam, India
shaik5moinuddin@yahoo.co.in

Abstract- A novel Incremental Processing method is proposed for data analysis in order to keep the mining results up-to-date. Data is continuously arriving by different data generating factors like social network, online shopping, sensors, e-commerce etc. Because of this Big Data the results of data mining applications getting stale and disused over time. Cloud intelligence applications often perform iterative computations (e.g., PageRank) on constantly changing data sets (e.g., Web graph). While previous studies extend MapReduce for efficient iterative computations, it is too expensive to perform an entirely new large-scale MapReduce iterative job to timelyaccommodate new changes to the underlying data sets. This paper, propose i2MapReduce to support incremental iterative computation. We observe that in many cases, the changes impact only a very small fraction of the data sets, and the newly iteratively converged state is quite close to the previously converged state. i2MapReduce exploits this observation to save re-computation by starting from the previously converged state, and by performing incremental up-dates on the changing data. The technique helps in improving the job running time and reduces the running time of refreshing the results of big data.

Keywords: Big data, Mining, Map reduce, Hadoop, MRBGraph.

I. INTRODUCTION

In typical data mining systems, the mining procedures require computational intensive computing units for dataanalysis and comparisons. A computing platform is, therefore, needed to have efficient access to, at least, twotypes of resources, data and computing processors. In Big Data mining, data scale is far beyond the capacitythat a single personal computer can handle, a typical Big Data processing framework will rely on cluster computers with a high-performance computing platform, with a data mining task being deployed by running some parallel programming tools, such as Map Reduce, on a large number of computing nodes. The role of thesoftware component is to make sure that a single data mining task, such as finding the best match of a query from a database with billions of records, is split into many small tasks each of which is running on one or multiple computing nodes. Such a Big Data system, which blends both hardware and software components, is hardly available without key industrial stockholder's support. In fact, for decades, companies have been making business decisions based on transactional data stored in relational databases.

Big Data mining offers opportunities to go beyond traditional relational databases to rely on less structured data weblogs, social media, e-mail, sensors, and photographs that can be mined for useful information. As Modern day Internet applications have created a need to manage immense amounts of data quickly. For example, devices and

ISSN 2454-9924 Volume: 7 Issue: 3(2016)

communication means like social networking sites, the amount of data produced by mankind is growing rapidly every year. It has become increasingly popular to mine such big data, which helps in taking business decisions or to provide better personalized good quality services. Big data technologies are important in providing more accurate analysis, which may lead to more concrete decision-making resulting in greater operational efficiencies, cost reductions, and reduced risks for the business. In many situations, it is desirable to periodically refresh the mining computation in order to keep the mining results up-to-date. Major business intelligence companies, such IBM, Oracle, Teradata, and so on, have all featured their own products to help customers acquire and organize these diverse data sources and coordinate with customer's existing data to find new insights and capitalize on hidden relationships.A large number of frameworks have been developed for big data analysis. MapReduce is one of the simple, generalized, framework used in production. Implementations of map-reduce enable many of the most common calculations on large-scale data to be performed on large collections of computers, efficiently and in a way that is tolerant of hardware failures during the computation. Here the main focus is on improving Map Reducetechnique. Incremental processing is an advanced approach to refreshing mining results. Given the size of the input big data, it is very heavy weighted to return the entire computation from scratch. Incrementally processing the new data of a large data set, takes state as implicit input and combines it with new data. MapReduce programming model is widely used for large scale and one-time data-intensive distributed computing, but it lacks for built-in support for the iterative process.

II. MAPREDUCE BACKGROUND

MapReduce is a one of a promising technique of computing that manage large-scale computations in a way that is tolerant of hardware faults. A MapReduce job usually partition the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. MapReduce includes two main functions, called Map and Reduce. MapReduce computation is shown in Figure 1. In the Figure 1 the system manages the parallel execution, coordination of a task that execute Map or Reduce, and also deals with the possibility that one of these tasks will fail to execute. These Map tasks turn the chunk into a sequence of key-value pairs<K, V>. The way key-value pairs are produced from the input data is determined by the code written by the user for the Map function. The key-value pairs from each Map task are composed by a master controller and sorted by key. The keys are divided among all the Reduce tasks, so all key-value pairs with the same key wind up at the same Reduce task. The Reduce tasks work on one key at a time and combine all the values associated with that key in some way. The manner of combination of values is determined by the code written by the user for the Reduce function.

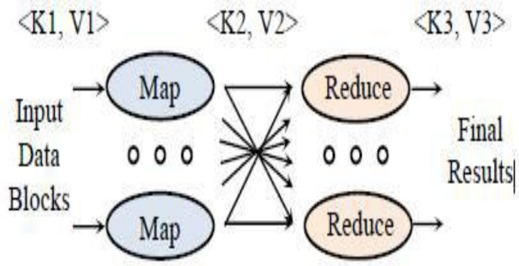

Figure 1. MapReduce computation

ISSN 2454-9924 Volume: 7 Issue: 3(2016)

III. LITERATURE SURVEY

1 MapReduce: Simplified Data Processing on Large Cluster

From the last five years, many authors and others at Google have implemented lots of special purpose computations that processes large amount of data such as web request logs, crawl data, etc. To compute varioustypes of derived data such as various representation of graph structure of web documents, most frequent queryin a day, etc. Many computations are straightforward. Most of the time input data is large. This data is distributed across many machines. In this system, design a new abstraction that allows to express the simplecomputation and trying to perform but hides the unstructured details of parallelism, data distribution and loadbalancing in library and fault tolerance. This abstraction is inspired by the map and reduce in many functionallanguages. In this map function key/value pairs are used. Apply reduce operation to all the values that sharedsame key, in order to combine the derived data appropriately.

2 Incoop: MapReduce for Incremental Computation

A computer system produces and collects increasing amounts of data. Services of Internet companies analyzing to improve services. Incoop system is generic framework which is based on Hadoop and use forincremental computation. Incoop can detects changes to the input data and enable the automatic updates of theoutput by reusing mechanism of fine-grain incremental processing. There are two case studies of higher-levelservices that are: i) Incremental query ii) Log Processing System without changing a single line of code ofapplication input data it improves the significant performance of results.

3 Big Data Mining using Map Reduce

Big data is large amount of data. Big data applications where data collection has grown continuously, itis expensive to capture, extract and manage and process data using existing software tools. For example, Forecasting of weather, Electricity Supply, Social media. With increasing size of data in data warehouse it isexpensive to perform data analysis. Data cube commonly abstract and summarize databases. It is way ofstructuring data in different n dimensions for analysis on some measure of interest. For data processing Big dataprocessing framework relay on cluster computers and parallel execution framework provided by Map-Reduce.

4 Iterative processing

A number of distributed frameworks have newly emerged for big data processing. HaLoop improves the efficiency of iterative computation by making the task scheduler loop-aware and by employing caching mechanisms. Twister employs a lightweight iterative MapReduce runtime system by sensibly constructing a Reduce-to-Map loop. IMapReduce supports iterative processing by directly passing the Reduce outputs to Map and by distinguishing variant state data from the static data.

5 Incremental processing for one-step application.

Besides Incoop, several recent studies aim at supporting incremental processing for one-step applications. Incoop detects changes to the inputs and enables the automatic update of the outputs by employing an efficient, fine-grained result reuse mechanism. This incremental nature of data suggests that performing large-scale computations incrementally can improve efficiency dramatically. But Incoop supports only task-level

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES
ISSN 2454-9924 Volume: 7 Issue: 3(2016)

incremental processing. So, Incoop do not allow for reusing the large existing base of MapReduceprograms.Incoop supports only one step computation.

6 Incremental processing for iterative application.

Naiad proposes a timely dataflow paradigm that allows stateful computation and arbitrary nested iterations. To support incremental iterative computation, programmers have to completely rewrite their MapReduce programs for Naiad. In comparison, we extend the widely used MapReduce model for incremental iterative computation. Existing Map-Reduce programs can be slightly changed to run on i2MapReduce for incremental processing

7 iMapReduce: A Distributed Computing Framework for Iterative Computation

Relational data pervasive in most of the applications such as a social network analysis and data mining.

These relational data containing at least millions and hundreds of relations. This need distributed computingframeworks for processing these data on large cluster. Example of such a framework is Mapreduce. This paperpresents iMapreduce, a framework that supports iterative processing. Users are getting allow by specified theiterative operations with map and reduce functions.

IV. PROBLEM DESCRIPTION

Many online data sets grow incrementally over time as new entries are slowly added and existing entries are deleted or modified. Taking advantage of this instrumentality, systems for incremental bulk data processing, such as Google‟s Percolator, can achieve efficient updates. This efficiency, however, comes at the price of losing compatibility with the simple programming models offered by non-incremental systems, e.g., Map Reduce, and more importantly, requires the programmer to implement application-specific dynamic/ incremental algorithms, ultimately increasing algorithm and code complexity. The task-level coarse-grain incremental processing system, Incoop, is not publicly available. Therefore, we cannot compare i2MapReduce with Incoop. Instead we compare i2MapReduce with existing MapReduce model on Hadoop.

4.1 Existing System

A number of previous studies have followed this principle and designed new programming models to support incremental processing. Unfortunately, the new programming models are drastically different from MapReduce, requiring programmers to completely re-implement their algorithms. On the other hand, Incoop extends MapReduce to support incremental processing. However, it has two main limitations. First, Incoop supports only *tasklevel*incremental processing. That is, it saves and reuses states at the granularity of individual Map and Reduce tasks. Each task typically processes a large number of key-value pairs (kv-pairs). If Incoop detects any data changes in the input of a task, it will rerun the entire task. While this approach easily leverages existing MapReduce features for state savings, it may incur a large amount of redundant computation if only a small fraction of kv-pairs havechanged in a task. Second, Incoop supports only *one-step* computation, while important mining algorithms, such as PageRank, require iterative computation. Incoop would treat each iteration as a separate MapReduce job.

ISSN 2454-9924 Volume: 7 Issue: 3(2016)

Disadvantages of Existing system:

1. The existing system cannot gives promising output which enough for working in the Big Data.
2. The update of any data will result in re-run the complete setup.
3. It does support only task-level incremental processing.
4. It does support only one-step computation.

4.2 PROPOSED SYSTEM: -

The proposed i2MapReduce, an extension to MapReduce that supports fine-grain incremental processing for both one-step and iterative computation. Compared to previous solutions, i2MapReduce incorporates the following three novel features:

1) Fine-grain incremental processing using MRBG-Store. Unlike Incoop, i2MapReduce supports kv-pair level fine-grain incremental processing in order to minimize the amount of recomputationasmuch as possible. The model the kv-pair level data flow and data dependence in a MapReduce computation as a bipartite graph, called MRBGraph.

2) General-purpose iterative computation with modest extension to MapReduce API.:our current proposal provides general-purpose support, including not only one-to-one, but also one-to-many,many-to-one, and many-to-many correspondence. Enhance the Map API to allow users to easily express loop-invariant structure data, and propose a Project API function to express the correspondence from Reduce to Map. While users need to slightly modify their algorithms in order to take full advantage of i2MapReduce.

3) Incremental processing for iterative computation. Incremental iterative processing is substantially more challenging than incremental one-step processing because even a small number of updates may propagate to affect a large portion of intermediate states after a number of iterations. To address this problem, this paper proposes to reuse the converged state from the previous computation and employ a change propagation control (CPC) mechanism. Also enhance the MRBG-Store to better support the access patterns in incremental iterative processing.

MRBG-Store

The MRBG-Store supports the preservation and retrieval of fine-grain MRBGraph states for incremental processing. User sees two main requirements on the MRBG-Store. First, the MRBG-Store must incrementally store the evolving MRBGraph. Consider a sequence of jobs that incrementally refresh the results of a big data mining algorithm. As input data evolves, the intermediate states in the MRBGraph will also evolve. It would be wasteful to store the entire MRBGraph of each subsequent job. Instead, user would like to obtain and store only the updated part of the MRBGraph. Second, the MRGB-Store must support efficient retrieval of preserved states of given Reduce instances. For incremental Reduce computation, i2MapReduce re-computes the Reduceinstance associated with each

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES
ISSN 2454-9924 Volume: 7 Issue: 3(2016)

changed MRBGraph edge.

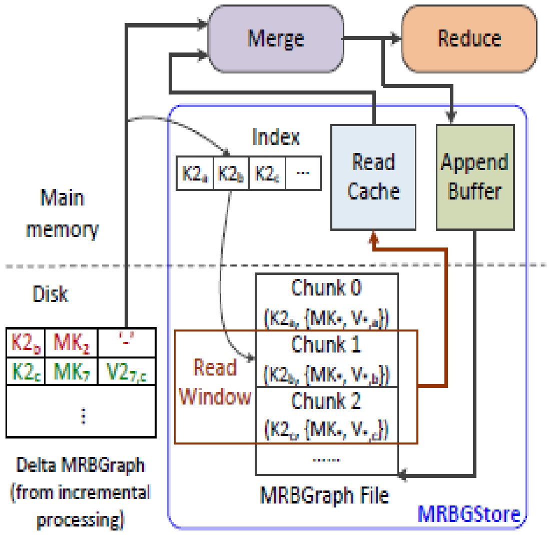

Figure 2: MRBG Store

For a changed edge, it queries the MRGB-Store to retrieve the preserved states of the in-edges of the associated K2, and merge the preserved states with the newly computed edge changes as shown in figure 2.

MRBG Architecture: -

Bipartite Graph: -a bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets **U** and **V** (that is, **U** and **V** are each independent sets) such that every edge connects a vertex in **U** to one in **V as** shown in figure 3. Vertex set **U** and **V** are often denoted as partite sets. Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles. The two sets **U** and **V** may be thought of as a colouring of the graph with two colours: if one colours all nodes in **U** blue, and all nodes in **V** green, each edge has endpoints of differing colours, as is required in the graph colouring problem. One often writes **G= (U, V, E**) to denote a bipartite graph whose partition has the parts **U** and **V**, with **E** denoting the edges of the graph. If a bipartite graph is not connected, it may have more than one bipartition; in this case, the **(U, V, E)** notation is helpful in specifying one particular bipartition that may be of importance in an application. If **|U|** and **|V|**, that is, if the two subsets have equal cardinality, then **G** is called a balanced bipartite graph. If all vertices on the same side of the bipartition have the same degree, then G is called biregular.

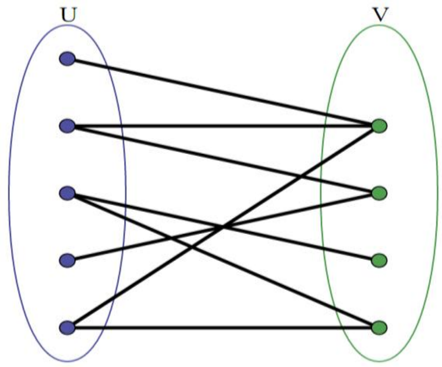

Figure 3: Bipartite Graph

MRBG Dataflow: -

MRBGraph Abstraction: User use a MRBGraph abstraction to model the data flow in MapReduce. Each vertex in the Map task represents an individual Map function call instance on a pair of (K1, V 1). Each vertex in the Reduce task represents an individual Reduce function call instance on a group of (K2, {V 2}). An edge from a Map instance to a Reduce instance means that the Map instance generates a (K2, V 2) that is shuffled to become part of the input to the Reduce instance. The input of Reduce instance a comes from Map instance 0, 2, and 4. MRBGraphedges are the fine-grain states M that user would like to preserve for incremental processing. An edge contains three pieces of information: (i) the source Map in Incremental data acquisition can significantly save the resources for data collection; it does not re-capture the whole data set but only capture the revisions since the last time that data was captured. (ii) the destination Reduce instance (as identified by K2), and (iii) the

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 7 Issue: 3(2016)

edge value (i.e. V 2). Since Map input key K1 may not be unique, i2MapReduce generates a globally unique Map key MK for each Map instance.

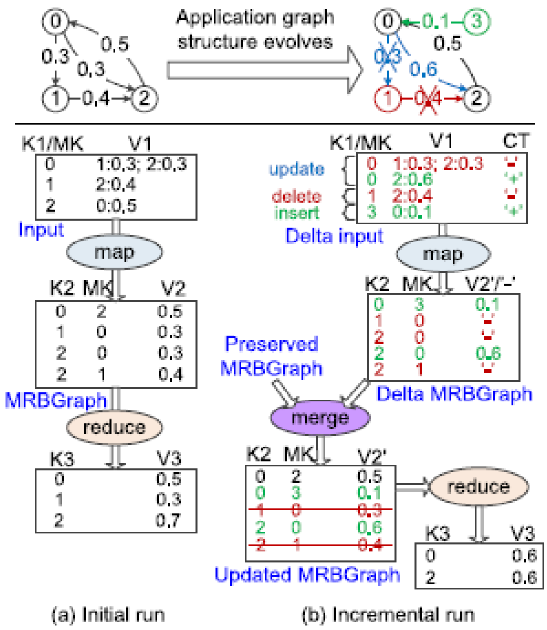

Figure 4: Data Flow of MRBGraph.

Therefore, i2MapReduce will preserve (K2, MK, V2) for each MRBGraph edge as shown in figure 4.

ALGORITHMS: -

Algorithm 1. Query Algorithm in MRBG-Store

Input queried key: k; the list of queried keys: L

Output chunk k

1: if !readcache.contains(k) then
2: gap <-0, w<- 0
3: i k's index inL //That is, Li = k
4: while gap < T and w + gap + length(Li) <read_cache:
size do
5: w <-w + gap +length(Li)
6: gap <-pos(Li+1)-pos(Li) – length(Li)
7: i<- i + 1
8: end while
9: starting from pos(k), read w bytes into read cache
10: end if
11: return read cache.get_chunk(k)

Algorithm 2. PageRank in MapReduce

PageRank is a well-known iterative graph algorithm for ranking web pages. It computes a ranking score for eachvertex in a graph. After initializing all ranking scores, the computation performs a MapReduce job per iteration. i and j are vertex ids, Ni is the set of out-neighbor vertices of i, Ri is i's ranking score that is updated iteratively. '|' means concatenation. All Ri's are initialized to one2. The Reduce instance on vertex j updates Rj by summing the Ri,j received from all its in-neighborsi, and applying a damping factor d.

Map Phase input: <i, Ni|Ri>
1: output <i, Ni >
2: for all j in Ni do
3: Ri,j = Ri |Ni|
4: output < j, Ri,j>
5: end for
Reduce Phase input: < j, {Ri,j ,Nj} >
6: Rj = dPiRi,j + (1 − d)
7: output < j, Nj |Rj>

Algorithm3. kmeans in MapReduce

Kmeans is a commonly used clustering algorithm that partitions points into k clusters. User denote the ID of a point as pid, and its feature values pval. The computation starts with selecting k random points as cluster centroids set {cid, cval}.As shown in Algorithm 3, in each iteration, the Map instance on a point pid assigns the point to the nearest centroid. The Reduce instance on a centroid cid updates the centroid by averaging the values of all assigned points {pval}.

Map Phase input: <pid, pval|{cid, cval} >
1: cid← find the nearest centroid of pval in {cid, cval}
2: output <cid, pval> Reduce Phase input: <cid, {pval} >

3: cval← compute the average of {pval}
4: output <cid, cval>

Algorithm4 GIM-V in MapReduce
Generalized Iterated Matrix-Vector multiplication (GIM-V) is an abstraction of many iterative graph mining operations.These graph mining algorithms can be generally represented by operating on an n × n matrix M and a vector v of size n. Suppose both the matrix and the vector are divided into sub-blocks. Let mi,j denote the (i, j)-th block of M and vj denote the j- th block of v. The computation steps are similar to those of the matrix-vector multiplication and can be abstracted into three operations: (1) mvi,j = combine2(mi,j ,vj); (2) v′i = combineAlli({mvi,j}); and (3) vi = assign(vi, v′ i).User can compare combine2 to the multiplication between mi,j and vj , and compare combineAll to the sum of mvi,j for row i. Algorithm 4 shows the MapReduce implementation with two jobs for each iteration. The first job assigns vector block vj to multiple matrix blocks mi,j (∀i) and performs combine2(mi,j , vj) to obtain mvi,j . The second job groups the mvi,j and vi on the same i, performs the combineAll({mvi,j}) operation, and updates vi using assign (vi , v′ i).

Map Phase 1 input:< (i, j),mi,j> or < j, vj>
1: if kv-pair is < (i, j),mi,j> then
2: output < (i, j),mi,j>
3: else if kv-pair is < j, vj> then
4: for all i blocks in j's row do
5: output < (i, j), vj>
6: end for
7: end if
Reduce Phase 1 input: < (i, j), {mi,j , vj} >
8: mvi,j = combine2(mi,j , vj)
9: output <i, mvi,j>, < j, vj>
Map Phase 2: output all inputs
Reduce Phase 2 input: <i, {mvi,j , vi} >
10: v′ i← combineAll({mvi,j})
11: vi ← assign(vi, v′ i)
12: output <i, vi >

Implementation
Proposed approach works in the following manner,
Step 1: Collection of evolving datasets
The evolving datasets will be collected for mapping and reduction.
Step 2: Development of mapping technique
The mapping techniques will be developed and the data will be mapped using these mapping techniques as shown in figure 5 using the concept of MRBG.
Step 3: Development of reduction technique
The reduction techniques will be developed and the mapped data will be reduced using these reduction techniques.
Step 4: Implementation on PageRank and K-means algorithm and GIM-V:
Step 5: Result Analysis and Comparison

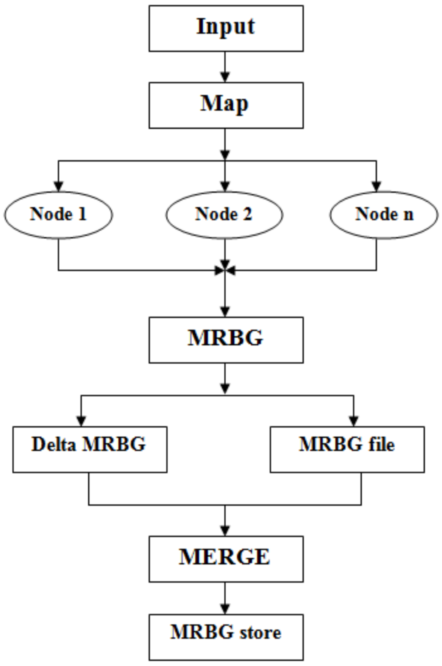

Figure 5: i^{2}MapReduce

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ENGINEERING TECHNOLOGIES

ISSN 2454-9924 Volume: 7 Issue: 3(2016)

1 Advantages of the Proposed System:

- To our knowledge, i2MapReduce is the *first MapReducebased solution that efficiently supports incremental iterative computation*.
- AMRBGStore is designed to preserve the fine-grain states in the MRBGraph and support efficient queries to retrieve fine-grain states for incremental processing.
- Unlike Incoop, i2MapReduce supports kv-pair level fine-grain incremental processing in order to minimize the amount of re-computation as much as possible.
- The current proposal provides general purpose support, including not only one-to-one, but also one-to-many, many-to-one, and many-to-many correspondence.
- User enhance the Map API to allow users to easily express loop-invariant structure data, and user propose a Project API function to express the correspondence from Reduce to Map.
- While users need to slightly modify their algorithms in order to take full advantage of i2MapReduce, such modification is modest compared to the effort to re-implement algorithms on a completely different programming paradigm.
- User propose to reuse the converged state from the previous computation and employ a change

propagation control mechanism.

2 Disadvantages of the Proposed System

Supporting Smaller Number of State kv-pairs. In some applications, the number of state keys is smaller than *n*. Kmeans is an extreme case with only a single state kv-pair. In these applications, the total size of the state data is typically quite small. Therefore, the backward transfer overhead is low. Under such situation, i2MapReduce does not apply the above partition functions. Instead, it partitions the structure kv-pairs using MapReduce‟s default approach, while replicating the state data to each partition.

V. EXPERIMENTAL RESULTS AND ANALYSIS

After implementing the programming models on three different raw data sets, the result obtained for k-means algorithm,PageRank algorithm and GIM-V, is depicted in figure6.it shows the performance analysis of i2mapreduce k-means algorithm,PageRank algorithm and GIM-Vi.e., shows the decrease in time consumed with respect to i2mapreduce model compared to that of MapReduce model.

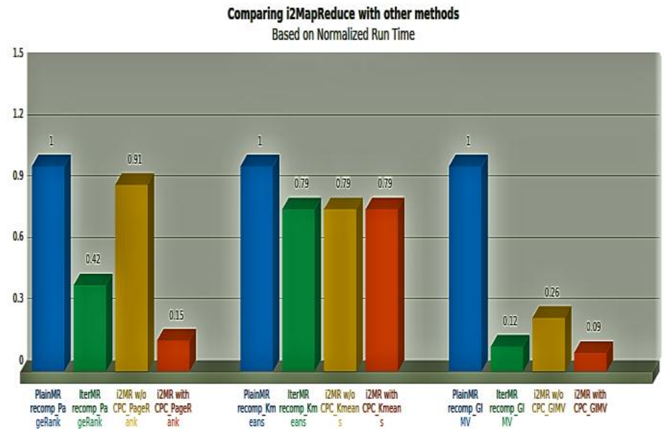

Figure 6: PageRank,K-means and GIM-V Algorithm Performance analysis using i^2MapReduce,

VI. Conclusion And Future Enhancement:

We have described I2MapReduce-Fine Grain Incremental Processing based on

ISSN 2454-9924 Volume: 7 Issue: 3(2016)

MapReduceframework. That supports k/v pair level fine-grain incremental processing to minimize the amount of recomputation and MRBG-Store to support efficient quires for retrieving fine-grain states for incrementalprocessing and to preserve the fine-grain states in MRBG This framework combines a fine-grain advanceengine, a general-purpose iterative model, and a set of effective techniques for fine-grain incremental iterativecomputation. Real-machine experiments show that I2MapReduce can significantly reduce the run time to refreshbig data mining result compared to re-computation on both plain and iterative MapReduce. We are studying cost-aware execution optimization that intelligently uses the MRBGraph state and selects the optimal executionstrategy based on online cost analysis.

In future, we are trying to identify the data changes whenever there is a dynamic updation using FP algorithm. Though retrieval of data becomes easier with map reduce, the interdependency of map and reduce tasks requires more fault tolerance. So, we are focusing on good fault tolerance solution by analysing and experimenting with various computing frameworks like pagerank, k-means,GIM-V etc.wepropse that array based languages,like R are ideal to express these algorithms for processing bigdata.

References

[1] Yanfeng Zhang, Shimin Chen, Qiang Wang, and Ge Yu, Member, IEEE, “i2MapReduce: Incremental MapReduce for Mining Evolving Big Data” IEEE Transactions On Knowledge And Data Engineering, Vol. 27, No. 7, July 2015.

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

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

[4] 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 Computing., 2012, pp. 534–541.

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

[6] 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 Computing., 2010, pp. 810–818.

[7] 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.

[8] 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 Computing., 2010, pp. 810–818.

[9] The R project for statistical computing. http://www.r-project.org.

[10] ShivaramVenkataraman,Indrajit Roy Alvin and Au Young Robert S. Schreiber,” Using R for Iterative and Incremental Processing,” in HotCloud'12 Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing Pages 11-11